UNIVERSITÉ DE MONTRÉAL

OPTIMISATION SANS DÉRIVÉES SOUS CONTRAINTES

MATHILDE PEYREGA
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

OPTIMISATION SANS DÉRIVÉES SOUS CONTRAINTES

présentée par : PEYREGA Mathilde
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. ORBAN Dominique, Doctorat, président
M. AUDET Charles, Ph. D., membre et directeur de recherche
M. LE DIGABEL Sébastien, Ph. D., membre et codirecteur de recherche
M. KOKKOLARAS Michael, Ph. D., membre
M. LUCET Yves, Ph. D., membre externe

DÉDICACE

À ma grand-mère

REMERCIEMENTS

J'ai le plaisir de remercier en premier lieu mes directeurs de recherche Charles Audet et Sébastien Le Digabel. Charles, ton cours de programmation mathématique m'a donné l'envie et la motivation pour entreprendre ce doctorat en optimisation. Sébastien, je suis fière d'être ta première étudiante au doctorat. Je te remercie d'avoir partagé avec moi tes tout premiers projets de recherche. Je vous remercie tout deux pour votre accompagnement tout au long de ce doctorat, pour avoir été toujours présents avec des pistes menant à des solutions, pour m'avoir encadrée et aussi pour m'avoir fait confiance pour cette dernière année où j'ai travaillé à distance. Je vous remercie pour tous vos conseils que je n'oublierai pas, en particulier ceux qui m'auront appris à présenter les travaux de recherche clairement. Je remercie aussi mon troisième collaborateur, Andy Conn, avec qui j'ai eu l'honneur de pouvoir travailler et rendre visite à IBM. Andy, c'est une chance extraordinaire pour moi d'avoir pu bénéficier de tes conseils expérimentés. Je remercie aussi Annie, Karine et Barbara, les épouses de ces trois chercheurs, pour les invitations pour des barbecues, des soupers, ainsi que pour m'avoir invitée lors de ma soutenance. Je n'oublie pas non plus Pierre Baptiste, alors directeur du département de MAGI, grâce à qui m'a orienté vers les chercheurs du Gerad lors de mon arrivée à Montréal. Merci à Miguel Anjos, professeur du département de MAGI, pour m'avoir conseillé lors de mes expériences d'enseignement à Polytechnique Montréal.

Je tiens à remercier les membres de mon jury, le professeur Yves Lucet qui a accepté de faire le déplacement pour ma soutenance, ainsi que les professeurs Dominique Orban et Michael Kokkolaras. Je remercie aussi Dominique pour les différentes discussions lors de mon doctorat, en particulier pour ses conseils en optimisation non linéaire.

Je voudrais remercier toutes les personnes qui m'ont enrichie durant ma thèse mais aussi mon séjour au Québec, en particuliers les étudiants du Gerad. Merci à Marilène, qui a été présente pour partager les bons moments comme les plus difficiles. Merci à Elspeth pour avoir fait équipe avec moi lors d'une compétition Mopta. Merci à Nadir pour son aide, en particulier lors de mon déménagement pour Vancouver, mais aussi pour son écoute. Merci aussi à Amina, Sara, Juan, Trish, Jesus, Stéphane, Vilmar, Adham, Jesús, et tous les autres, pour tout le temps passé avec vous. Merci à tous les étudiants du Gerad qui ont participé aux séminaires pas ordinaires, et ont eu à coeur d'échanger sur leur recherche. Je voudrais aussi remercier le personnel administratif du Gerad, dont Carole et Marie, qui ont été une aide et un soutien discret mais important lors de mon doctorat.

Je voudrais enfin remercier ma famille. Je commence par ma mère, grâce à qui j'ai pu faire de belles études. Maman, merci pour ton soutien et ta patience, de l'autre côté de l'Atlantique. Merci

à ma soeur, Marie, qui m'a permis d'étudier à Paris, et qui a tant pris soin de moi. Merci à mon frère Mathieu qui m'a le premier donné le goût des mathématiques. Enfin, je voudrais remercier mon mari, Pierre, qui m'a épaulé, soutenu et aimé durant ces dernières années. Merci aussi pour tous tes conseils techniques qui m'ont aidé à développer et débugguer les différents codes de mon doctorat. Notre fils et toi m'avez tout deux donné l'énergie pour finir cette thèse.

RÉSUMÉ

L'optimisation sans dérivées (*Derivative-Free Optimization*, DFO) et l'optimisation de boîtes noires (*Blackbox Optimization*, BBO) est un champ de la recherche opérationnelle en pleine extension, qui correspond à de nouveaux problèmes pour lesquels toutes les fonctions en jeu ou seulement une partie ne sont pas connues analytiquement mais sont le résultat d'expériences ou de simulations numériques, appelées communément boîtes noires. Les contraintes peuvent être de différentes natures. Elles peuvent être connues analytiquement ou bien elles peuvent être, comme la fonction objectif, le résultat de la boîte noire. Elle peuvent même être ignorées de l'utilisateur qui les découvre malgré lui, alors qu'il cherche à évaluer la boîte noire en un point qu'il pensait être réalisable. Elles peuvent être lisses ou non lisses. Cette thèse s'intéresse plus particulièrement aux traitements des contraintes dans le cadre de l'optimisation sans dérivées et de l'optimisation de boîtes noires. Il s'agit donc de proposer de nouvelles techniques pour résoudre des problèmes sous contraintes.

Tout d'abord, une méthode générique de traitement des égalités linéaires est proposée. Différents convertisseurs sont utilisés afin de reformuler le problème initial en un problème réduit dans le sous-espace affine défini par les égalités linéaires. Différentes stratégies combinant en plusieurs étapes ces convertisseurs sont proposées. Une implémentation de cette technique dans un algorithme de recherche directe, MADS, utilisant le logiciel NOMAD, est réalisée. À partir de tests numériques, une stratégie est retenue. Elle surpasse sur les problèmes testés un autre logiciel de recherche directe, HOPSPACK, qui proposait déjà un traitement spécifique des contraintes d'égalités linéaires. De plus, notre méthode est adaptable à tous les algorithmes existants.

Ensuite, un algorithme hybride, combinant des outils issus de l'optimisation sans dérivées, basée sur les modèles, et ceux de l'optimisation de boîtes noires, basée sur des méthodes de recherche directe, est proposé à travers un algorithme de région de confiance sans dérivées (*Derivative-Free Trust-Region*, DFTR) qui revisite la barrière progressive déjà proposée dans MADS, et qui permet de traiter certains types de contraintes d'inégalités. L'algorithme obtenu offre des résultats compétitifs avec un représentant de l'optimisation sans dérivées, COBYLA, et un représentant de l'optimisation de boîtes noires, MADS, à partir de tests réalisés sur un panel de problèmes académiques mais aussi sur deux boîtes noires issues de l'optimisation multidisciplinaire.

Enfin, un dernier algorithme sans dérivées a été développé, afin de pouvoir résoudre des problèmes avec des contraintes générales d'égalités ou d'inégalités, et qui utilise une méthode classique de Lagrangien augmenté. L'algorithme utilisant le Lagrangien augmenté sert à résoudre le sous-problème de la région de confiance mais aussi à définir les règles de mise à jour de l'algorithme. Des résultats sur des problèmes académiques permettent de conclure quant à la validité de la méthode.

ABSTRACT

Derivative-Free Optimization (DFO) and Blackbox Optimization (BBO) are growing optimization fields. The goal is to handle new problems involving functions for which analytical expressions are not explicit, but which are the results of simulations or experiments, called blackboxes. Different kind of constraints can be encountered. Their analytical expressions can be given, or they can be the result of the blackbox. They can even be hidden, and not known by the user. They can be smooth or nonsmooth. This thesis focuses more specificaly on the contraints in DFO and BBO, and its goal is to develop new techniques to solve constrained problems.

First, a generic method for linear equalities is proposed. Different converters are used to reformulate the initial problem into a reduced one, in the subspace defined by the linear equalities. Different strategies combining these converters in multi-step algorithms are proposed. Theses techniques are implemented in a direct-search algorithm, MADS, by using the solver NOMAD. Computational tests allow to choose the best strategy with the best results. On a benchmark of analytical problems our algorithm outperforms a direct-search algorithm implemented in the solver HOPSPACK, which also handles directly linear equalities. The proposed method is transposable to any other DFO or BBO algorithm.

Then, a derivative-free trust-region (DFTR) algorithm combining DFO tools, based on models, and BBO tools, based on direct-search techniques, is proposed through a (DFTR) algorithm. The progressive barrier first designed for MADS is revisited and allows to solve general inequalities in this new DFTR algorithm. The new algorithm offers competitive results with COBYLA, a DFO software and NOMAD, a BBO software. Computational experiments are conducted on a set of analytical problems and two blackboxes from multidisciplinary design optimization.

Finally, a third DFO algorithm is proposed, allowing to solve equality and inequality constrained problems, by using an augmented Lagrangian method. This one is used to solve the trust-region subproblem but also to design simple update rules for the DFTR algorithm. Computational results on analytical problems validate our method.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

DFO	Derivative-Free Optimization
CS	Coordinate Search
GPS	Generalized Pattern Search
DFTR	Derivative-Free Trust-Region
BBO	Blackbox Optimization
MADS	Mesh Adaptive Direct Search
PBTR	Progressive Barrier derivative-free Trust-Region

CHAPITRE 1 INTRODUCTION

Il existe des problèmes d'optimisation qui ne comportent que quelques dizaines de variables et qui pourtant posent des difficultés majeures à résoudre, car la fonction objectif et les contraintes sont le résultat de simulations ou encore d'expériences. En effet, certains systèmes en ingénierie ne peuvent pas être modélisés directement par un modèle mathématique explicite. Ces simulations ou expériences sont associées à des boîtes noires, qui prennent en entrée des valeurs des variables et retournent des valeurs de sortie. Les valeurs de sortie peuvent être la fonction objectif et les contraintes. Chaque évaluation de la boîte noire peut prendre du temps ou être coûteuse. Le budget des évaluations permises en est donc limité.

Les méthodes classiques d'optimisation utilisent les dérivées des fonctions représentant l'objectif et les contraintes du problème. Mais dans le cas de problèmes dont les fonctions sont les sorties d'une boîte noire, il est impossible d'avoir accès aux dérivées des fonctions. Les fonctions peuvent même être non dérivables, c'est ce qu'on appellera les fonctions *non lisses*, pas opposition aux fonctions *lisses*, qui elles sont différentiables.

Les techniques de différentiation automatique ou toutes autres techniques pour estimer directement les dérivées ne peuvent pas s'appliquer, car elles sont trop coûteuses en terme d'évaluations, ou simplement inapplicables lorsque les fonctions représentées par la boîte noire ne peuvent pas être approximées par des fonctions lisses.

Différents algorithmes ont été développés dès les années 1950, avec la recherche par coordonnées [51] afin de mettre en œuvre des techniques qui n'utilisent pas les dérivées des fonctions à optimiser. Un renouveau des premiers algorithmes a eu lieu dans les années 1990, et de nouvelles techniques ont supplanté les anciennes. Dans ces dernières années, l'optimisation sans dérivées et l'optimisation de boîtes noires a connu un essor important comme évoqué dans la récente enquête [12] ou dans [23].

Les applications sont déjà nombreuses. On compte, entre autres, des applications dans le domaine aéronautique, avec la conception d'un rotor d'hélicoptère [30, 31] ou l'optimisation de formes d'ailes pour réduire le bruit aérodynamique [78]. Il existe aussi des applications dans le domaine spatial, avec la conception d'un bouclier thermique [63, 1]. Le domaine hydroélectrique et l'hydrologie n'est pas oublié, avec les applications présentées dans [53], [7] et [82]. Des problèmes de géométrie moléculaire sont aussi améliorées [8] grâce à l'optimisation de boîtes noires.

Plusieurs classifications des problèmes rencontrés existent, et différentes dénominations cohabitent pour évoquer ces problèmes issus principalement de simulations ou d'expériences. Différents cher-

cheurs comme Audet et Kokkolaras [23] choissent de distinguer les différents champs de l'optimisation sans dérivées en fonction de la nature du problème à traiter. Dans la suite de ce document, le terme boîte noire ne représentera pas uniquement un système qui prend en entrée des variables et retourne des sorties. Les boîtes noires désignent la forme la plus générique de problèmes, dont la structure des fonctions objectif et contraintes ne peut être exploitée. En pratique, les fonctions en jeu ne sont pas dérivables, et sont hautement complexes. Elles peuvent être bruitées, ne pas être définies en certains points sans que les contraintes soient clairement identifiées, et elles peuvent contenir des variables de catégorie qui influencent, selon leur valeur, la définition même du problème et sa dimension réelle. Les contraintes et leur nature peuvent aussi être difficiles. L'existence même d'une contrainte peut être ignorée. Une nomenclature complète des contraintes a été récemment proposée par Le Digabel et Wild [68]. Les dérivées de la fonction objectif ou des contraintes d'une boîte noire sont considérées comme inexistantes, les fonctions en jeu sont non lisses.

L'optimisation sans dérivées quant à elle désigne le fait de seulement utiliser l'évaluation des fonctions objectif et contraintes, sans avoir recours à leurs dérivées. Cependant, par opposition à l'optimisation de boîtes noires, les fonctions en jeu sont simplement approximables par des fonctions lisses.

Ainsi, nous désignerons par BBO, pour *Blackbox Optimization*, l'optimisation de boîtes noires, qui s'intéresse aux problèmes dont les dérivées sont inexistantes, et par DFO, pour *Derivative-Free Optimization*, l'optimisation sans dérivées des problèmes dont les dérivées existent mais sont inaccessibles. Les algorithmes BBO pourront s'appliquer aux problèmes DFO et vice-versa dans certains cas particuliers.

Les méthodes développées s'appuient sur des analyses de convergence qui permettent de garantir la qualité des solutions proposées dans le cas où les fonctions seraient Lipschitz ou différentiables. Cependant, les méthodes d'optimisation BBO ou DFO, qui n'exploitent pas les dérivées ne doivent pas être utilisées lorsque les gradients sont simplement calculables ou estimables. Ces analyses de convergence sont simplement une preuve de la solidité des algorithmes. Elles permettent de distinguer ces algorithmes des méthodes heuristiques, comme par exemple les algorithmes génétiques.

Traditionnellement, DFO et BBO, l'optimisation sans utilisation des dérivées des fonctions, est classifiée en deux grandes familles d'algorithmes. La première famille est celle des algorithmes DFO basés sur des modèles, qui approximent localement les fonctions de la boîte noire. Les algorithmes BBO, la deuxième famille, utilisent la plupart du temps des techniques de recherche directe, qui consistent en des sondes dans certaines directions.

Les traitements des contraintes varient selon le statut orienté DFO ou orienté BBO d'un algorithme. Les algorithmes BBO permettent actuellement de traiter les contraintes les plus difficiles, allant jusqu'aux contraintes cachées, ignorées de l'utilisateur. Les algorithmes DFO permettent un traite-

ment plus rapide des problèmes, mais seules certaines contraintes aux caractéristiques particulières sont traitées. Les algorithmes BBO ne tirent que rarement profit des informations connues sur les contraintes, et inversement les algorithmes DFO ne traitent que rarement les contraintes de boîtes noires.

Cette thèse, rédigée par articles, a pour objectif de proposer des méthodes de gestion des contraintes pour les problèmes BBO et DFO. Lorsque les contraintes sont linéaires, il est important de tirer profit des informations disponibles pour réduire la dimension des problèmes. Les problèmes BBO et DFO présentant des similitudes, les méthodes de gestion de contraintes BBO peuvent être adaptées aux méthodes DFO la plupart du temps. Inversement, les méthodes classiques d'optimisation avec dérivées peuvent aussi être adaptées aux algorithmes DFO.

Le premier chapitre présente une revue de littérature des principales méthodes BBO et DFO, ainsi que des analyses de convergence associées et des méthodes de gestion des contraintes pour chaque famille de méthodes. Différents types de contraintes seront distingués. Le deuxième chapitre expose la démarche de recherche et l'organisation de la recherche réalisée. Les trois articles sont présentés dans les trois chapitres suivants. Enfin, une discussion générale et une conclusion permettent de synthétiser le travail effectué, mais aussi de présenter les limites et les perspectives de la recherche mise en oeuvre.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente les problèmes DFO et BBO, les différents types de contraintes, ainsi que les principaux algorithmes d'optimisation qui n'utilisent pas les dérivées des fonctions afin de résoudre le problèmes DFO et BBO sans ou avec contraintes.

2.1 Présentation du problème d'optimisation

2.1.1 Forme générale du problème

Qu'il soit DFO ou BBO, le problème mono-objectif le plus générique rencontré est le suivant :

$$\min_{x \in \Omega} f(x) \tag{2.1}$$

avec $f : \mathbb{R}^n \to \mathbb{R} \cup \left\{\infty\right\}$ la fonction objectif et $\Omega \subseteq \mathbb{R}^n$ l'ensemble réalisable.

La difficulté principale de (2.1) est de trouver le meilleur point tout en contrôlant le nombre d'évaluations de la fonction objectif et des fonctions impliquées dans la définition de Ω. La notion de budget d'évaluations apparaît, car chaque évaluation peut-être coûteuse.

Dans la suite, nous présentons différents type de contraintes définissants l'ensemble Ω.

Aucune hypothèse n'est faite a priori sur les fonctions en jeu, que ce soit f ou les fonctions qui seront utilisées pour définir l'ensemble réalisable Ω. Certains considèrent même, pour les problèmes BBO, que les fonctions en jeu peuvent ne pas être de vraies fonctions, car deux évaluations en un même point peuvent renvoyer des valeurs différentes à cause d'un bruit stochastique par exemple.

2.1.2 Différent type de contraintes

Nous reprenons ici les termes définis dans la nomenclature proposée dans [68].

L'ensemble réalisable Ω peut être décomposé de la manière suivante :

$$\Omega = \{x \in X \ : \ c^{\mathcal{E}}(x) = 0 \text{ et } c^{\mathcal{I}}(x) \leq 0\},$$

avec $c^{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^m \cup \left\{\infty\right\}$ et $c^{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^p \cup \left\{\infty\right\}$ les contraintes d'égalités et d'inégalités, et X un sous-ensemble de $\mathbb{R}^n$.

Les fonctions $c^{\mathcal{E}}$ et $c^{\mathcal{I}}$ définissent des fonctions connues, l'utilisateur sait qu'elles existent, par opposition aux constraintes cachées [36] que l'utilisateur découvre alors qu'il essaie, en vain, d'évaluer les fonctions en un point qu'il pensait réalisable. Elles sont aussi soit a priori, soit le résultat d'une simulation. Une contrainte a priori est explicitée analytiquement, on connait son expression mathématique. Lorsque ces contraintes sont le résultat de simulation, elles sont le résultat de simulation soit de type BBO, soit de type DFO. Elles peuvent être lisses ou non, approximables par des fonctions lisses ou non.

L'ensemble X représente un ensemble de points satisfaisant des contraintes connues ou cachées mais non-relaxables, c'est-à-dire que si ces contraintes ne sont pas satisfaites, alors les autres fonctions objectif et contraintes ne peuvent être évaluées. Cet ensemble X peut être par exemple défini par des contraintes de bornes, qui sont connues, ou bien par des contraintes cachées. Les contraintes cachées sont toujours le résultat de simulation.

Enfin, une contrainte peut être quantifiable ou non. Lorsqu'une contrainte est non-quantifia-ble, nous pouvons uniquement connaître son statut, réalisé ou violé. Lorsqu'une contrainte est quantifiable, nous avons accès à une mesure qui indique la distance par rapport au domaine réalisable. Dans cette thèse, les contraintes $c^{\mathcal{E}}$ et $c^{\mathcal{I}}$ sont toujours quantifiables et relaxables. Si une contrainte est non-quantifiable, nous considérerons qu'elle fait partie des contraintes définissant l'ensemble X, mais l'ensemble X peut aussi être défini avec des contraintes quantifiables, comme les contraintes de bornes.

Les problèmes DFO se caractérisent par le fait qu'ils ne contiennent que des contraintes connues (a priori ou de simulation) et quantifiables, et que les contraintes connues de simulation sont approximables par des fonctions lisses.

Les problèmes BBO se caractérisent par le fait qu'au moins une des fonctions, qu'elle définisse l'objectif ou une contrainte, est non approximable par une fonction lisse, ou bien qu'il existe au moins une contrainte cachée ou une contrainte non quantifiable.

Les problèmes DFO et BBO peuvent tous deux avoir des contraintes non relaxables. C'est le cas par exemple des contraintes de bornes qui peuvent être non relaxables.

2.1.3 Différentes familles d'algorithmes pour différentes familles de problèmes

Nous avons vu que les différents problèmes peuvent être classés en deux familles, DFO et BBO, et que différentes complexités dans les contraintes y sont associées.

Dans ce contexte, plusieurs familles d'algorithmes sont utilisées selon que les problèmes soient du type DFO ou BBO. Différentes techniques de traitement des contraintes existent qu'on soit dans le cadre DFO ou dans le cadre BBO. Chaque algorithme contient une preuve de convergence. Cette

preuve permet de garantir que, pour des fonctions relativement lisses ou présentant des propriétés particulières, l'algorithme converge vers un minimum local quelque soit le point initial. Même si en pratique on ne peut pas savoir si une boîte noire vérifiera les hypothèses permettant d'appliquer l'analyse de convergence, de telles preuves permettent de garantir mathématiquement la qualité de l'algorithme.

Les méthodes dédiées aux problèmes BBO sont les méthodes de recherche directe, qui évaluent à chaque itération la boîte noire en un ensemble de points, définis au début de l'itération. Ces méthodes reposent principalement sur des techniques de sonde. Les principaux représentants de ces méthodes sont CS, GPS, MADS, implémentés dans le logiciel NOMAD ([2]), mais aussi GSS ([64]) implémenté dans HOPSPACK ([87]). Les méthodes traitant les contraintes sont des méthodes dites de barrière extrême, de filtre, et de barrière progressive. Dans le cas où les contraintes sont approximables par des fonctions lisses, un algorithme de Lagrangian augmenté est aussi adapté à l'algorithme GSS.

Les méthodes spécialisées pour les problèmes DFO sont les méthodes basées sur les modèles. Les principaux représentants de cette famille de méthodes sont les algorithmes de région de confiance sans dérivées, en anglais *Derivative-Free Trust-Region*, soit DFTR. Comme les fonctions en jeu pour les problèmes DFO sont simplement approximables par des fonctions lisses, il est alors intéressant de chercher de bons modèles pour approximer le problème DFO, et ensuite appliquer des méthodes adaptées des algorithmes classiques d'optimisation. Une théorie a été développée pour construire de bons modèles. Les méthodes de traitement des contraintes développées jusqu'à maintenant sont des adaptations des méthodes d'optimisation classiques avec dérivées.

Les principaux logiciels implémentant ces méthodes sont DFO de Scheinberg, et les logiciels développés par Powell, dont par exemple COBYLA ([89]) et LINCOA ([92]).

Nous ne présentons pas ici les méthodes qui distinguent l'évaluation des contraintes et de la fonction objectif, comme expliqué par exemple dans [79].

Nous n'évoquerons pas non plus les méthodes heuristiques qui ne possèdent pas d'analyse de convergence.

2.2 Méthodes de recherche directe

Les algorithmes de recherche directe étudiés dans le cadre de cette thèse sont les suivants :

- Recherche par coordonnées (*Coordinate Search* en anglais, abrégé par CS) [51].
- Algorithme GPS (*Generalized Pattern Search*) [100].
- Algorithm GSS (*Generating Set Search*) [64].
- Algorithme MADS (*Mesh Adaptive Direct Search*) [17].

Ces méthodes ont été développées dans cet ordre. Nous reprenons ici en partie la présentation de ces algorithmes faite dans [21].

2.2.1 Recherche par coordonnées et barrière extrême

Présentation de l'algorithme

L'algorithme CS est itératif. Il résout des problèmes sans contraintes ou avec des contraintes de bornes (section 2.2.1). On l'itinialise avec un point $x_0 \in \mathbb{R}^n$ et un pas $\Delta_0 \in \mathbb{R}_+$. A l'itération k, on effectue une recherche locale autour de l'itéré courant $x_k \in \mathbb{R}^n$, appelée *sonde* (*poll* en anglais), dans $2n$ directions (les n directions de base et les n directions opposées), à la recherche d'un point dont la valeur est plus petite que celle de l'itéré courant. L'ensemble des points de sonde, P_k, est défini par $P_k := \{x_k \pm \Delta_k e_j : j \in \{1, 2, ..., n\}\}$, où les e_j sont les vecteurs de la base et Δ_k le pas à l'itération k. En cas d'échec, on diminue la taille du pas après la sonde. En cas de succès, la sonde permet de déterminer le nouvel itéré courant. L'algorithme est résumé à la figure 2.1.

En pratique, plusieurs conditions d'arrêt peuvent être choisies : un pas inférieur à un paramètre Δ_{min}, un nombre maximum d'évaluations de f (budget d'évaluations), un nombre maximum d'itérations, un nombre maximum d'itérations sans améliorations de l'objectif, *etc*.

$[0]$ **initialisation**

  $x_0 \in \mathbb{R}^n$, $\Delta_0 \in \mathbb{R}_+$, $k \in \mathbb{Z}$
  $k \leftarrow 0$

$[1]$ **recherche locale (*sonde*)**

  $P_k \leftarrow \left\{x_k \pm \Delta_k e_j \ , \ \ j = 1, \ldots, n\right\}$
  si $(\exists\, t \in P_k \ : \ f(t) < f(x_k))$
    $x_{k+1} \leftarrow t$
    $\Delta_{k+1} \leftarrow \Delta_k$
  sinon (x_k optimum local pour P_k)
    $x_{k+1} \leftarrow x_k$
    $\Delta_{k+1} \leftarrow \Delta_k/2$
  si (condition d'arrêt rencontrée) : **STOP**
  $k \leftarrow k + 1$
  aller en $[1]$

Figure 2.1 Algorithme de recherche par coordonnées.

Variantes possibles : Différentes stratégies de recherche

Plusieurs améliorations simples sont possibles, en particuliers au niveau de l'ordre dans lequel on évalue les points de P_k. Voici les différentes stratégies que l'on peut appliquer :

- **Stratégie complète** : On évalue tous les points de P_k, et on retient celui qui a la plus petite valeur de f, si cette valeur est inférieure à celle de l'itéré courant.
- **Stratégie opportuniste** : A l'itération k, on peut ne pas évaluer f pour tous les points de P_k, mais cesser les évaluations de l'itération dès qu'un meilleur point t a été trouvé.
- **Stratégie ordonnée** : À priori, l'examen des points de P_k se fait selon l'ordre des directions selon lesquelles ces points ont été générés. Cet ordre peut être modifié de manière dynamique : si à l'itération k le point t est tel que $f(t) < f(x_k)$ et qu'on a $x_{k+1} \leftarrow t$, alors la direction utilisée pour générer t est placée en tête de la liste des directions pour la prochaine itération. On arrête la sonde dès que l'on a trouvé un point améliorant.

Résultat de convergence globale

On appelle convergence globale le fait qu'une méthode de minimisation, quel que soit le point initial choisi, converge en un point $\hat{x}$ qui soit un minimum local.

Le résultat de convergence suivant a été établi pour l'algorithme de recherche par coordonnées dans le cas sans contraintes [15] :

Proposition 1 *Dans le cas sans contraintes, si $\hat{x}$ est la limite d'une suite de points en lesquels* CS *connaît un échec, et si f est continûment différentiable sur un voisinage de $\hat{x}$, alors $\nabla f(\hat{x}) = 0$.*

Traitement des contraintes de bornes avec CS : barrière extrême

La méthode utilisée pour traiter les contraintes de bornes est la méthode de la barrière extrême :

On définit $f_X : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ de la façon suivante :

$$f_X(x) = \begin{cases} f(x) \text{ si } x \in X, \\ +\infty \text{ sinon.} \end{cases}$$

Dans l'algorithme, un point hors de Ω ne pourra jamais être choisi comme itéré courant. C'est pour cela que le point initial x_0 doit être choisi dans Ω. D'un point de vue pratique, il n'est pas nécessaire d'évaluer toutes les contraintes si jamais il s'avère qu'une contrainte déjà évaluée est violée. Cette approche peut être utile quand l'évaluateur de la contrainte ne donne pas de mesure de la réalisabilité, c'est-à-dire lorsque la contrainte n'est pas quantifiable.

Definition 1 *Une contrainte quantifiable est une contrainte pour laquelle on connaît la mesure de la violation de la contrainte.*

Nous avons le résultat suivant de convergence :

Proposition 2 *Soit $\hat{x}$ limite d'une suite de points en lesquels* CS *connaît un échec. Si la suite des itérés produits par* CS *est bornée, et si f est strictement différentiable près de $\hat{x}$, alors $f^\circ(\hat{x}; d) \geq 0 \; \forall d \in T_X^F(\hat{x})$ (où $T_X^F(\hat{x})$ est le cône des directions réalisables).*

Exemple critique

Malgré ces différentes stratégies, et les garanties de convergence globale dans le cas où la fonction est continûment différentiable sur $\mathbb{R}^n$, la méthode de recherche par coordonnées n'est pas toujours efficace. Voici un exemple qui montre bien ses limites : l'exemple de la Figure 2.2 consiste à minimiser la fonction $f : x \to \|x\|_\infty$ avec $x \in \mathbb{R}^2$. Le point optimal est $x^* = [0\,0]^T$ avec $f(x^*) = 0$. Cependant, si l'algorithme est initialisé avec un point x_0 tel que $x_0 \neq x^*$ et $\left|x_0^T e_1\right| = \left|x_0^T e_2\right|$, alors toutes les itérations seront des échecs. En effet, $\forall \Delta > 0$ et $\forall j \in \{1, 2\}$, on a $f(x_0 \pm \Delta e_j) \geq f(x_0)$. Le problème est que f n'est pas différentiable : on n'a pas de garantie que l'algorithme converge à un point stationnaire. Ici, l'algorithme de recherche par coordonnées converge à un point pour lequel le gradient n'est pas défini, mais pour lequel il existe des directions de descente.

Deux inconvénients de la recherche par coordonnées sont ainsi mis en évidence par cet exemple :

- Le nombre des directions possibles pour la sonde se limite aux $2n$ directions $\pm e_i$ ($i \in \{1, 2, ..., n\}$).
- On n'utilise qu'une recherche locale. Aucune stratégie de recherche au niveau global n'est essayée. Pourtant, sur notre exemple, il aurait été facile de trouver des points générés au hasard meilleurs que $x_0 = [1\,1]^T$).

2.2.2 Algorithme GPS et filtre

L'algorithme *Generalized Pattern Search*, ou *recherche par motifs*, a été proposé en 1997 par Torczon et al. [100], puis reformulé par [32] dans le format GPS dont il est question ici. GPS est une évolution de l'algorithme de recherche par coordonnées : il permet un plus grand nombre de directions de recherche, une plus grande flexibilité dans leur choix, ainsi qu'une recherche à un niveau global. Pour un historique complet des méthodes ayant mené aux algorithmes de type GPS, voir [64].

Il permet de résoudre des problèmes sans contraintes, et des problèmes avec des contraintes linéaires connues a priori comme nous le verrons à la section 2.2.2.

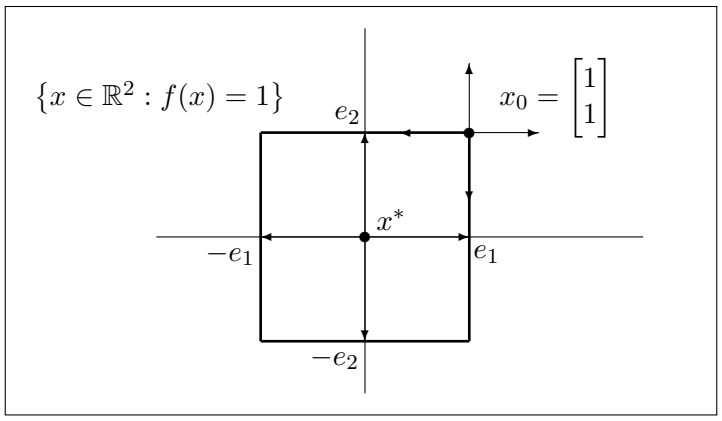


Figure 2.2 Recherche par coordonnées avec $f(x) = \|x\|_\infty$. En gras, la courbe de niveau $f = 1$. Extrait de [21].

Description de l'algorithme

Afin d'introduire cet algorithme, nous aurons besoin des notions de base positive et d'ensemble générateur positif définies par Davis [46].

Definition 2 *Un* ensemble générateur positif *pour $\mathbb{R}^n$ est un ensemble fini de vecteurs tel que leurs combinaisons linéaires positives engendrent $\mathbb{R}^n$.*

Definition 3 *Une* base positive *est un ensemble générateur positif minimal au sens de l'inclusion. C'est-à-dire qu'aucun sous-ensemble strict de ces vecteurs n'est un ensemble générateur positif.*

L'algorithme est présenté à la figure 2.3. L'ensemble des directions possibles sont les colonnes d'une matrice D qui doit être de la forme $D = GZ$ avec $G \in \mathbb{R}^{n\times n}$ une matrice non singulière et $Z \in \mathbb{Z}^{n\times q}$, tandis que l'ensemble des directions à l'itération k, $D_k = \left\{d_k^1, d_k^2, \ldots, d_k^{q_k}\right\} \subseteq D$ doit être un ensemble générateur positif (on a $n+1 \leq q_k = |D_k| \leq q = |D|$). Une itération k est qualifiée de *succès* si on a trouvé un point $t \in M_k$ tel que $f(t) < f(x_k)$, et sinon l'itération est un *échec*. A l'itération k, V_k est l'ensemble des points où la fonction f a été évaluée complètement au début de l'itération et le treillis (*mesh*) est l'ensemble ainsi défini :

$$M_k = \bigcup_{x\in V_k} \left\{x + \Delta_k D z \ : \ z \in \mathbb{N}^q\right\}, \tag{2.2}$$

et l'ensemble des points de sonde possibles (le cadre) s'écrit

$$P_k = \left\{x_k + \Delta_k d \ : \ d \in D_k\right\} \subseteq M_k \, . \tag{2.3}$$

La condition pour la mise à jour du paramètre de treillis Δ_k est la suivante :

$$\Delta_{k+1} = \tau^{\omega} \Delta_k \tag{2.4}$$

avec $\tau \in \mathbb{Q}$ et ω entier fini, positif si l'itération k est un succès et strictement négatif sinon.

La stratégie de recherche globale est laissée à la discrétion de l'utilisateur, ce qui apporte une grande flexibilité à l'algorithme : en effet, il peut alors utiliser sa connaissance du problème pour tâcher de trouver des points prometteurs. Les seules contraintes sont que ces points soient en nombre fini et appartiennent au treillis courant M_k.

On peut remarquer que CS est un cas particulier de GPS avec $D = [I \ -I]$, $\tau = 2$, $\omega = 0$ en cas de succès et $\omega = -1$ sinon. Les stratégies vues à la section 2.2.1 sont encore applicables ici (la stratégie optimiste est désormais incluse dans l'algorithme de base grâce aux paramètres τ et ω). Abramson *et al.* [3] ont décrit des améliorations supplémentaires possibles lorsque l'on possède de l'information même incomplète sur les dérivées de f : dans l'étape de sonde, par exemple, si le signe des dérivées partielles au centre de sonde est connu, il sera possible d'élaguer les directions de sonde à une seule direction.

[0] **initialisation**

- x_0, Δ_0, V_0, D
- $k \leftarrow 0$

[1] **recherches globale et locale**

- **si** (condition d'arrêt rencontrée) : **STOP**
- **recherche globale** (optionnelle)
  - évaluer f sur un ensemble fini $S_k \subseteq M_k$
- **recherche locale** (*sonde*)
  - choisir $D_k \subseteq D$ ensemble générateur positif
  - évaluer f sur $P_k \subseteq M_k$ (définition 2.3)

[2] **mises à jour**

- mettre à jour V_k
- **si** $\left(\exists\, t \in V_k : f(t) < f(x_k)\right)$ (*succès*)
  - $x_{k+1} \leftarrow t$
- **sinon** (*échec*)
  - choisir $u \in V_k$ tel que $f(u) = f(x_k)$
  - $x_{k+1} \leftarrow u$
- mettre à jour Δ_{k+1} en respectant (2.4)
- $k \leftarrow k + 1$, aller en [1]

Figure 2.3 Algorithme GPS.

Ainsi, Δ_k est à la fois la taille du treillis mais aussi la taille du pas.

Exemple de cadres et de treillis pour GPS

La figure 2.4 donne un exemple de cadres possibles dans $\mathbb{R}^2$ avec différentes valeurs de Δ_k.

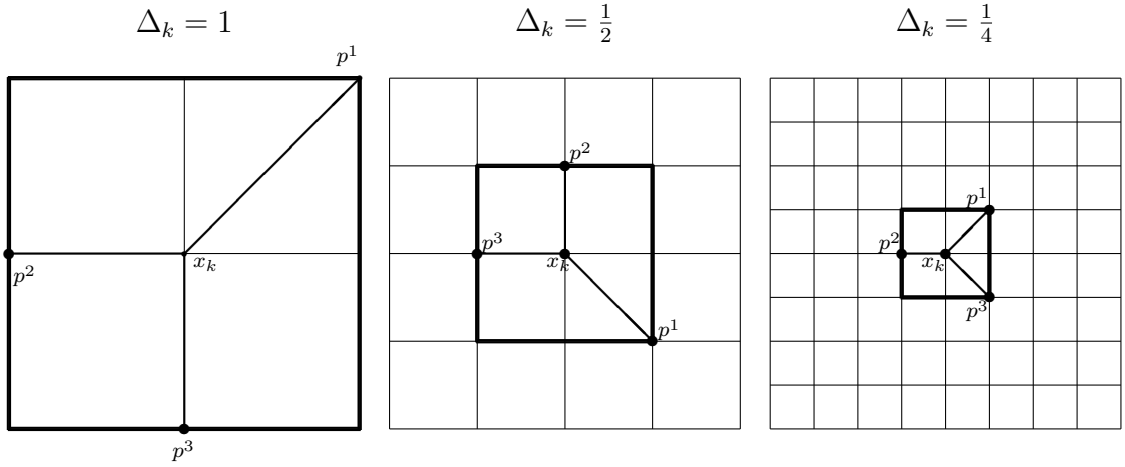


Figure 2.4 Exemple de cadres de GPS $P_k = \{x_k + \Delta_k d : d \in D_k\} = \{p^1, p^2, p^3\}$ pour différentes valeurs de Δ_k. Dans les trois figures, le treillis M_k est l'intersection des lignes et des colonnes. Extrait de [17].

Différences avec CS

GPS est un algorithme supérieur à CS puisqu'il le généralise. Les trois principales différences de GPS par rapport à CS sont les suivantes :

— Plusieurs autres directions sont possibles que les $2n$ directions de l'algorithme CS.
— Avant l'étape de sonde, on introduit une étape de recherche globale, qui permet à l'utilisateur d'exploiter sa connaissance du problème
— L'algorithme permet l'augmentation de la taille du pas en cas de succès à une itération, ce qui permet d'augmenter significativement la vitesse de convergence.

Résultat de convergence dans le cas sans contraintes

On a quelques résultats de convergence avec GPS. Nous allons en citer un, dans le cas sans contraintes. Mais avant cela, nous introduisons deux définitions :

Definition 4 *Une sous-suite convergente $\{x_k\}_{k\in K}$ (pour un sous-ensemble d'indice K) est dite* raffinante *si* $\lim_{k\in K} \Delta_k = 0$.

On utilise dans cette section la définition de la dérivée généralisée de Clarke.

Definition 5 *Pour une fonction f Lipschitz près de $\hat{x}$, la* dérivée généralisée de Clarke *de f en $\hat{x}$ dans la direction $v \in \mathbb{R}^n$ est :*

$$f^\circ(\hat{x}; v) = \limsup_{\substack{y \to \hat{x} \\ t \searrow 0}} \frac{1}{t} \Big(f(y + tv) - f(y) \Big) . \tag{2.5}$$

Definition 6 *f est strictement différentiable en $\hat{x}$, d'après la définition de Clarke [37], si f est différentiable en $\hat{x}$ et si*

$$\lim_{\substack{y \to \hat{x} \\ t \searrow 0}} \frac{1}{t} \Big(f(y + tv) - f(y) \Big) = \nabla f(\hat{x})^T v \quad \forall v \in \mathbb{R}^n .$$

Proposition 3 *Dans le cas non contraint, si $X = \mathbb{R}^n$, et sous l'hypothèse que la suite des itérés générés par* GPS *est bornée, alors d'après [15] on a les résultats suivants :*

- *Si $\hat{x}$ est limite d'une sous-suite raffinante, et si f est strictement différentiable près de $\hat{x}$, alors $\nabla f(\hat{x}) = 0$.*
- *Si $\hat{x}$ est limite d'une sous-suite raffinante, et si f est Lipschitz près de $\hat{x}$, alors $f^\circ(\hat{x}; d) \geq 0$ pour toute direction d qui est utilisée pour une infinité d'itérations.*

Audet [11] a montré que ces résultats de convergence sont les meilleurs possibles dans le sens où toutes les hypothèses sont nécessaires pour garantir ces résultats de convergence.

Traitement des contraintes avec GPS : contraintes d'inégalité

Contraintes de bornes et contraintes linéaires

Lewis et Torczon, dans [70], présentent une modification de GPS qui permet de traiter les contraintes de bornes.

Les mêmes auteurs présentent une solution pour traiter les contraintes linéaires connues a priori avec GPS, dans [71]. Cette méthode généralise la précédente, puisqu'une contrainte de borne est un cas particulier de contraintes linéaires. À chaque itération, si x_k est l'itéré courant, l'ensemble D_k est choisi de manière à contenir les directions qui engendrent les cônes $T_\Omega^F(y)$ pour tous les points $y \in B_\epsilon(x_k)$ (boule ouverte de centre x_k et de rayon ϵ), comme l'illustre la figure 2.5.

Cette méthode converge globalement sous certaines hypothèses, dont certaines qui imposent que f soit continuement différentiable.

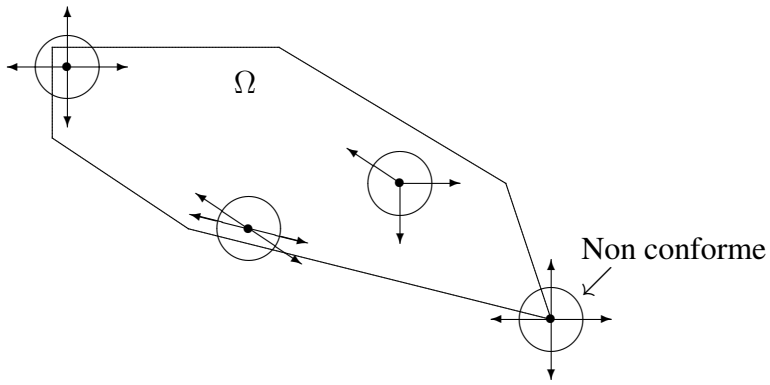


Figure 2.5 Exemples de trois ensembles de directions de sonde qui sont colinéaires aux contraintes qui définissent le domaine, et une quatrième qui ne respecte pas cette condition. Extrait de [21].

Méthode du filtre

Pour GPS, une méthode a été développée pour traiter les contraintes d'inégalités. Cette méthode a été développée par Fletcher et Leyffer [52], puis adaptée par Audet et Dennis [16]. Nous n'entrerons pas dans les détails de cette méthode, puisqu'une méthode plus élaborée a été développée à partir de la méthode du filtre : c'est la méthode de la barrière progressive [17]. Elle est applicable à la méthode GPS, mais elle a été développée pour MADS : voir section 2.2.3.

Exemple critique

Voici un exemple extrait de [64] où l'algorithme GPS ne donne pas un résultat satisfaisant. Soit $f(x) = \Big(1 - exp(-\|x\|^2)\Big) \times \max\Big\{\|x-c\|^2, \|x-d\|^2\Big\}$, $x \in \mathbb{R}^2$ et $c = -d = [30\ 80]^T$ (courbes de niveau représentées à la figure 2.6). L'optimum est $x^* = [0\ 0]^T$ pour $f(x^*) = 0$. La fonction f est Lipschitz et strictement différentiable près de x^*. Trois essais sont effectués, avec $x_0 = [-3.3\ 1.2]^T$ et $\forall k$

- $D_k = D_1 = \{e_1, e_2, -e_1, -e_2\}$ ***ou***
- $D_k = D_2 = \Big\{[1\ 0]^T, [0\ 1]^T, \Big[-\sqrt{2}/2 \quad -\sqrt{2}/2\Big]^T\Big\}$ ***ou***
- $D_k = D_3 = \Big\{[1\ 0]^T, \Big[-1/2 \quad \sqrt{3}/2\Big]^T, \Big[-1/2 \quad -\sqrt{3}/2\Big]^T\Big\}$

(directions représentées à la figure 2.7).

Chaque essai converge vers $\hat{x} = [-3.2\ 1.2]^T$. Mais $\hat{x}$ n'est pas un optimum. On peut constater, sur la Figure 2.6, qu'au point $\hat{x}$, aucune direction de descente ne peut être générée par l'algorithme GPS. Cet exemple met en évidence, de la même manière qu'en 2.2.1, que le fait d'utiliser un nombre fini de directions peut conduire à de mauvais résultats. La fonction f n'est ni strictement différentiable en $\hat{x}$, ni Lipschitz en $\hat{x}$: on n'a donc aucune garantie de convergence globale. D'autres exemples

pathologiques sont décrits dans [11], dont même un cas où GPS converge à un point où le gradient existe et est non nul.

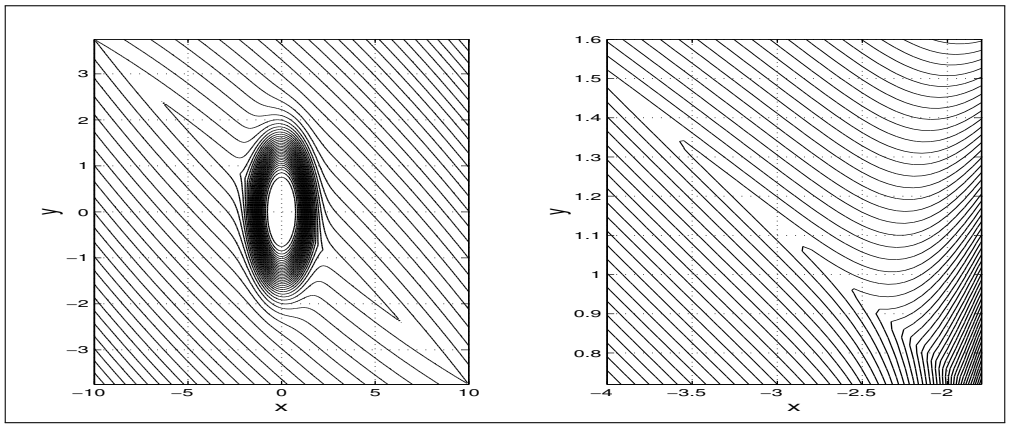


Figure 2.6 Courbes de niveau de $f(x) = \Big(1 - exp(-\|x\|^2)\Big) \times \max\Big\{\|x - c\|^2, \|x - d\|^2\Big\}$.

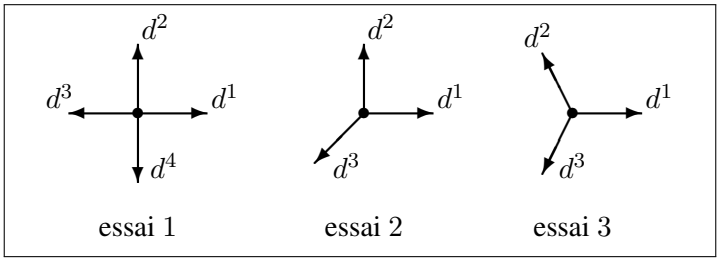


Figure 2.7 Directions utilisées pour minimiser f avec GPS.

Algorithme GSS et Méthode du Lagrangien augmenté

Les auteurs de [64] présentent la définition de l'algorithme GSS comme étant une généralisation de l'algorithme GPS avec la possibilité d'introduire une condition de décroissante minimale de la fonction objectif (*sufficient decrease* en anglais). Pour traiter des contraintes explicites générales, Lewis et Torczon [72] propose une méthode de pénalité pour traiter les contraintes h et g à l'aide d'un Lagrangian augmenté. La preuve de convergence suppose que les fonctions $c^{\mathcal{E}}$ et $c^{\mathcal{I}}$, qui peuvent être données par simulation, doivent être C^2.

2.2.3 Algorithme MADS et barrière progressive

L'algorithme MADS a été proposé par Audet et Dennis [17]. Il est une généralisation de l'algorithme GPS auquel il apporte un ensemble de directions de sonde normalisés dense dans la sphère

unité de $\mathbb{R}^n$.

Description de l'algorithme

La différence majeure avec GPS est que nous considérons désormais deux paramètres : Δ_k^m pour le treillis et Δ_k^p pour la sonde (alors que dans GPS : $\Delta_k = \Delta_k^m = \Delta_k^p$). A l'itération k, V_k est l'ensemble des points où la fonction f a été évaluée au début de l'itération, et on redéfinit le treillis de la sorte :

$$M_k = \bigcup_{x \in V_k} \left\{ x + \Delta_k^m Dz \ : \ z \in \mathbb{N}^p \right\} .$$

L'itéré courant x_k est également appelé *centre de sonde*, qui devient :

$$P_k = \left\{ x_k + \Delta_k^m d \ : \ d \in D_k \right\} \subseteq M_k \ .$$

Les conditions pour les directions sont les suivantes :

- $D = GZ \in \mathbb{R}^{n \times p}$ doit être un ensemble générateur positif, avec $G \in \mathbb{R}^{n \times n}$ non singulière et $Z \in \mathbb{Z}^{n \times p}$.
- À l'itération k, D_k doit aussi être un ensemble générateur positif.
- Les limites (telles que définies dans Coope et Price [44]) des ensembles D_k normalisés sont des ensembles générateurs positifs.
- $\forall d \in D_k \ , \ \exists \ u \in \mathbb{N}^p$ tel que $d = Du$.
- La distance entre x_k et un point $x_k + \Delta_k^m d$ de P_k ($d \in D_k$) est bornée :

$$dist(x_k, x_k + \Delta_k^m d) = \Delta_k^m \|d\| \leq \Delta_k^p \max \left\{ \|d'\| : d' \in D \right\} . \tag{2.6}$$

Une autre différence importante avec GPS est que D_k n'est pas nécessairement inclus dans D.

Les paramètres de treillis et de sonde doivent vérifier

$$\begin{cases} \Delta_k^m \leq \Delta_k^p \quad \forall k, \\ \lim\limits_{k \in K} \Delta_k^m = 0 \Leftrightarrow \lim\limits_{k \in K} \Delta_k^p = 0 \quad \forall K \text{ ensemble infini d'indices .} \end{cases} \tag{2.7}$$

L'algorithme est présenté à la figure 2.8.

Exemples de cadres et de treillis pour MADS

La Figure 2.9 extraite de [17] donne un exemple de cadres possibles dans $\mathbb{R}^2$ avec différentes valeurs de Δ_k^m. Nous pouvons d'ailleurs remarquer que la figure 2.4 est un exemple de GPS, mais

[0] initialisation

$x_0 \in \mathbb{R}^n$
$D \in \mathbb{R}^{n \times p}$
$\Delta_0^m, \Delta_0^p \in \mathbb{R}$
$k, k_{max}, \omega^+ \in \mathbb{N}$
$\omega^- \in \mathbb{Z}$, $\omega^- \leq -1$
$\tau \in \mathbb{Q}$, $\tau > 1$
$k \leftarrow 0$

[1] recherche globale

choisir $S_k \subseteq M_k$
si $(\exists\, t \in S_k \ :\ f(t) < f(x_k))$ (*succès*)
  choisir ω dans $\{0, 1, \ldots, \omega^+\}$
  aller en [3]

[2] sonde locale

$D_k \leftarrow \{d_k^1, d_k^2, \ldots, d_k^{s_k}\}$ (en fonction de Δ_k^p et tel que les conditions pour les directions sont vérifiées)
$P_k \leftarrow \{x_k + \Delta_k^m d\ ,\ d \in D_k\} \subseteq M_k$
si $(\exists\, t \in P_k \ :\ f(t) < f(x_k))$ (*succès*)
  choisir ω dans $\{0, 1, \ldots, \omega^+\}$
sinon (x_k *minimum local du cadre, échec*)
  choisir t dans V_k tel que $f(t) = f(x_k)$
  choisir ω dans $\{\omega^-, \omega^- - 1, \ldots, -1\}$
si (condition d'arrêt rencontrée) : **STOP**

[3] mises à jour

$x_{k+1} \leftarrow t$
$\Delta_{k+1}^m \leftarrow \tau^\omega \Delta_k^m$
$\Delta_{k+1}^p \leftarrow$ fonction de Δ_k^p telle que (2.7) est vérifiée
mettre à jour V_k
$k \leftarrow k + 1$
aller en [1]

Figure 2.8 Algorithme MADS.

aussi de MADS avec $\Delta_k = \Delta_k^m = \Delta_k^p$.

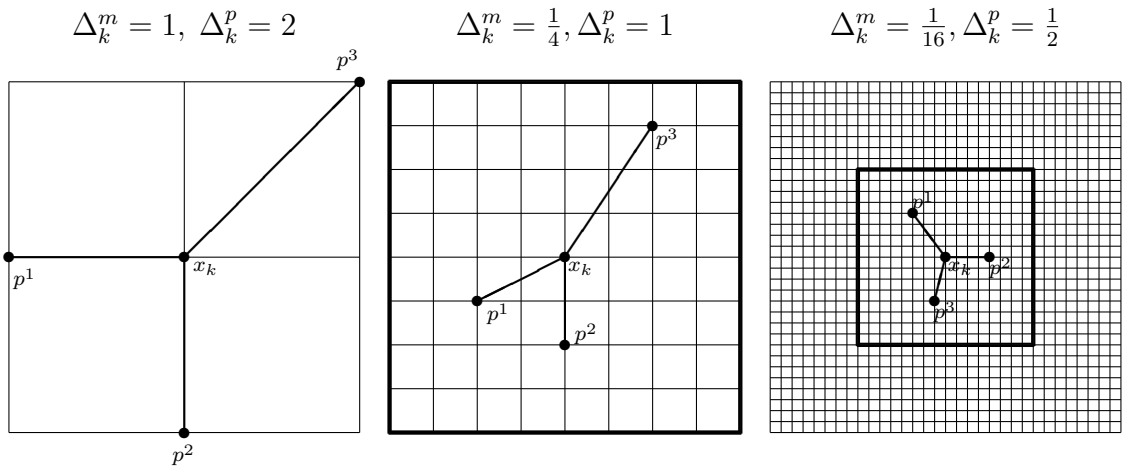


Figure 2.9 Exemples de cadres de MADS $P_k = \{x_k + \Delta_k^m d : d \in D_k\} = \{p^1, p^2, p^3\}$ pour différentes valeurs de Δ_k^m and Δ_k^p. Dans les trois figures, le treillis M_k est l'intersection des lignes horizontales et verticales. On remarque que le nombre de points de sonde possibles augmente de façon exponentielle quand Δ_k^m décroît.

Résultat de convergence dans le cas sans contrainte

Ce qui fait la force de MADS est le fait qu'un ensemble de directions dense est généré par l'algorithme. Ainsi, toutes les directions sont approchables. C'est ce qui offre les résultats de convergence sur lesquels nous reviendrons dans la section 2.2.3.

Traitement des contraintes avec MADS : la barrière progressive

La barrière progressive est une méthode développée pour traiter les contraintes d'inégalités relaxables et quantifiables dans MADS. La méthode de la barrière extrême, et celle du filtre, avait comme défaut de chercher trop rapidement un point réalisable, au détriment de la qualité de la fonction objectif. Cette nouvelle méthode, quant à elle, évite cet écueil en poussant la recherche de manière moins rapide vers le domaine réalisable. Ainsi, cela permet de trouver un point réalisable par un chemin constitué de points non réalisables tout en améliorant la qualité de l'objectif.

Barrière progressive

On définit la fonction de violation des contraintes comme pour les méthodes de filtres proposées en premier par Fletcher et Leyffer [52] :

$$h(x) = \begin{cases} \sum\limits_{i \in I} \max(c_i^{\mathcal{I}}(x), 0)^2 \text{ si } x \in X, \\ \infty \text{ sinon .} \end{cases}$$

I est ici l'ensemble des indices des contraintes d'inégalités. Chaque contrainte $c_i^{\mathcal{I}}$ est relaxable et quantifiable, pour tout $i \in I$.

On peut remarquer que h est une fonction non-négative, et que $x \in \Omega$ si et seulement si $h(x) = 0$. On aurait aussi pu choisir une autre norme que la norme 2 pour définir h.

On peut aussi remarquer que la barrière progressive joue le rôle d'une barrière extrême pour l'ensemble X défini par les contraintes non relaxables ou non quantifiables.

On utilise la notion de point dominant :

Definition 7 *Un point* $x \in \mathbb{R}^n$ domine $y \in \mathbb{R}^n$ *(noté* $x \prec y$*) si et seulement si* $f(x) \leq f(y)$, $h(x) \leq h(y)$ *et* $\Big(f(x), h(x)\Big) \neq \Big(f(y), h(y)\Big)$.

Résumé de l'algorithme Le point initial x_0 doit être choisi dans X. Puis, à chaque itération, il y a un ou deux centres de sonde :

- Un centre de sonde réalisable, si l'on a déjà trouvé un point réalisable. Les valeurs de f et h en ce point sont f_k^F et h_k^F.
- Un centre de sonde non réalisable. Les valeurs de f et h en ce point sont f_k^I et h_k^I.

On réalise une sonde autour de ces deux centres. Chaque nouvelle évaluation d'un point y est classée dans trois catégories :

- Succès total si $h(y) = 0$ et $f(y) \leq f_k^F$, ou si $h(y) > 0$ et $f(y) < f(x)\ \forall x \in I_k$ où $I_k \in \arg\min\limits_{x \in U_k} \{f(x) \ : \ 0 < h(x) < h_k^{max}\}$ avec U_k l'ensemble des points non dominés.
- Succès partiel si $0 < h(y) < h_k^I$ et $f(y) > f_k^I$.
- Échec si $h(y) = 0$ et $f(y) \geq f_k^F$, ou si $h(y) = h_k^I$ et $f(y) \geq f_k^I$, ou si $h(y) > h_k^I$.

Plusieurs stratégies d'exploration autour des centre réalisable et irréalisable peuvent être choisies. h_{max} est un seuil qui définit une barrière, et l'on ne considère que les points y tels que $h(y) \leq h_{max}$.

Le seuil h_{max} est alors réduit ou gardé constant en fonction de la catégorie de l'itération. Tous les points à droite de h_{max} ne sont plus pris en compte.

Remarque 1 *Cette méthode a l'avantage, par rapport à la barrière extrême (et à la méthode du*

filtre), de continuer à effectuer des sondes autour d'un point non réalisable, qui satisfait la barrière, et qui a un meilleur f que le centre de sonde réalisable. La barrière extrême impose un traitement trop brutal qui permet de trouver un point réalisable assez rapidement, mais à partir de ce point il est parfois coûteux voire impossible d'améliorer la valeur de f.

Les figures 2.10 et 2.11 illustrent l'algorithme. Sur la première, on voit l'image du domaine $X \subset \mathbb{R}^2$ dans l'espace des fonctions h v.s. f. Dans la deuxième, on effectue une sonde autour de l'itéré principal x^F. On ne génère pas de point qui soit un succès parmi les points réalisables, mais on génère un point non-réalisable qui est un succès partiel, et devient notre nouveau point non réalisable principal.

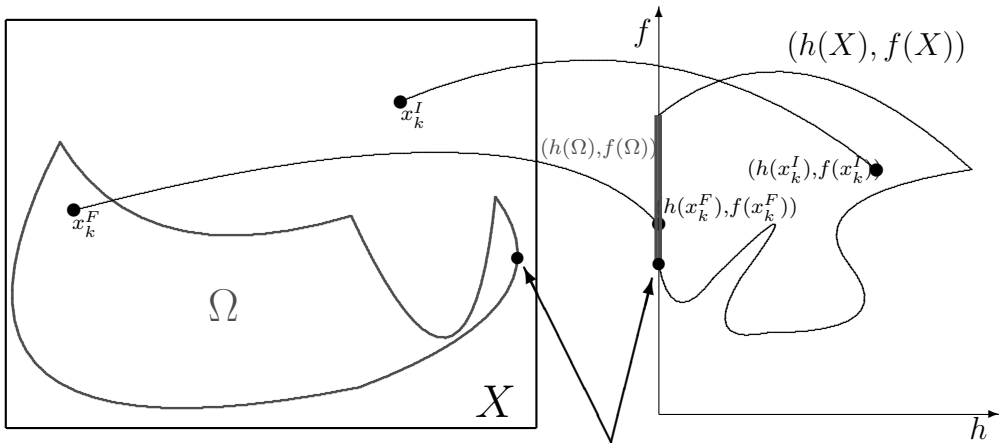


Figure 2.10 Le domaine réalisable Ω et le domaine X du problème d'optimisation, ainsi que leurs images dans l'espace des fonctions h vs f. Extrait de [20].

Résultats de convergence pour MADS **avec la barrière progressive**

Nous résumons ici les principaux résultats de convergence de l'algorithme MADS démontrés par Audet et Dennis [17].

Proposition 4 *Tout d'abord, les paramètres Δ_k^p et Δ_k^m vérifient*

$$\liminf_{k\to\infty} \Delta_k^p = \liminf_{k\to\infty} \Delta_k^m = 0 \, .$$

Definition 8 *En un point $\hat{x} \in \Omega$, le cône hypertangent $T_\Omega^H(\hat{x})$ est l'ensemble des vecteurs $v \in \mathbb{R}^n$ tels qu'il existe $\epsilon \geq 0$ vérifiant :*

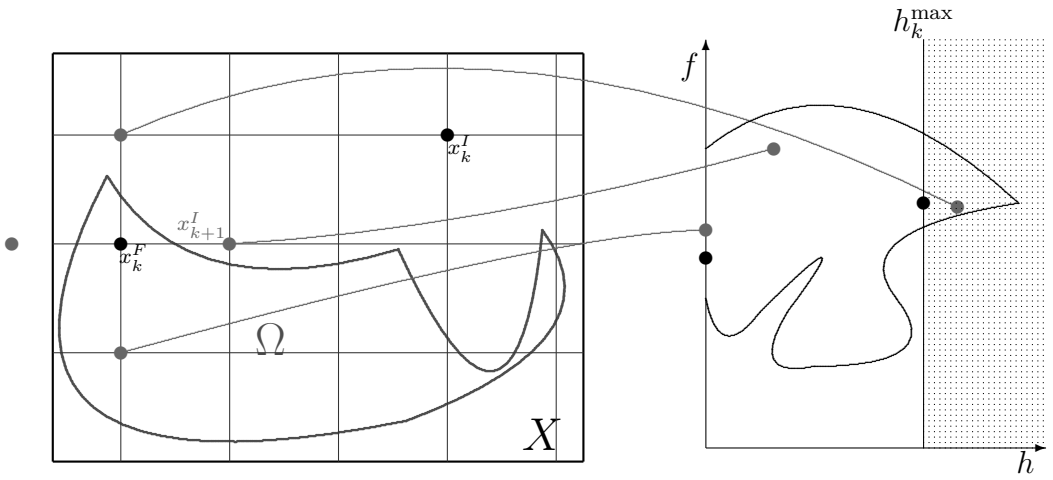


Figure 2.11 Une sonde autour de l'itéré principal réalisable x^F génère un nouvel itéré principal non réalisable x_{k+1}^I. Extrait de [20].

— $y + tw \in \Omega$, $\forall y \in \Omega \cap B_\epsilon(x)$,
— $w \in B_\epsilon(v)$,
— $0 < t < \epsilon$.

Proposition 5 *Soit f une fonction Lipschitz dans un voisinage de $\hat{x} \in \Omega$, où $\hat{x}$ est limite d'une sous-suite raffinante. Soit $v \in T_\Omega^H(\hat{x})$ une direction raffinante pour $\hat{x}$. Alors $f^\circ(\hat{x}; v) \geq 0$.*

Ce résultat nous indique que, dans le cas où la fonction possèdent certaines propriétés, alors on a convergence globale en un point qui est un point critique de Clarke. Mais ce n'est pas forcément un optimum local.

Les exemples lisses que nous connaissons jusqu'à maintenant pour lesquels l'algorithme MADS converge en un point critique de Clarke qui n'est pas un minimum local sont des exemples très artificiels. Cette preuve de convergence offre un certificat de garantie de la méthode MADS bien que nous ne connaissons rien aux propriétés des fonctions des boîtes noires qui seront optimisées par ce biais.

2.3 Méthodes de région de confiance basées sur les modèles : méthodes DFTR

Nous allons présenter les principes généraux des algorithmes de région de confiance sans dérivées, les méthodes DFTR (*Derivative-free trust-region methods* en anglais), proposées dans [42] et [43].

2.3.1 Construction des modèles par interpolation et régression

Les modèles les plus utilisés sont ceux construits par interpolation ou régression à partir d'un échantillon de points. Cet échantillon de points peut être sous-déterminé, complètement déterminé, ou sur-déterminé. En ces points, les valeurs des fonctions f, $c^{\mathcal{E}}$ et $c^{\mathcal{I}}$ sont connues. Cet échantillon de points est noté $\mathcal{Y}^k$ à l'itération k, et autour du point x.

Modèles semblables aux polynômes de Taylor d'ordre 1 et 2

Afin de conserver des modèles de qualité similaire à ceux construits avec les dérivées, Conn, Scheinberg et Vicente [43, chap. 6] proposent les définitions suivantes s'appliquant aux modèles présentant une analogie entre les modèles construits et les polynômes de Taylor d'ordre 1 et 2.

Definition 2.3.1 *Soit m_f un modèle de $f \in \mathcal{C}^1$ en $x \in \mathbb{R}^n$, soit $\Delta > 0$ un rayon. Supposons qu'il existe une constante positive κ telle que pour tout $y \in B(x;\Delta)$ le modèle m_f satisfait :*

$$\begin{aligned} \|\nabla f(y) - \nabla m_f(y)\| &\leq \kappa\Delta, \\ \|f(y) - m_f(y)\| &\leq \kappa(\Delta)^2. \end{aligned}$$

*Alors le modèle m_f est dit être un modèle κ-pleinement-linéaire (κ-*fully-linear *en anglais) de f sur $B(x;\Delta)$.*

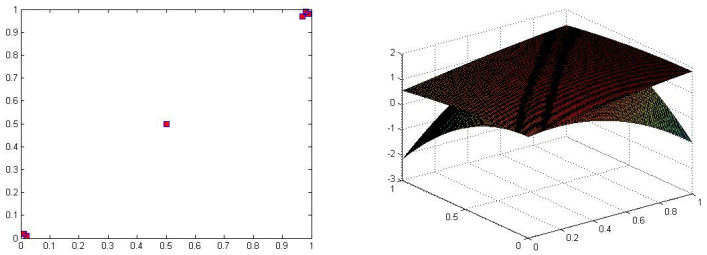
Definition 2.3.2 *Soit m_f un modèle de $f \in \mathcal{C}^2$ en $x \in \mathbb{R}^n$, soit $\Delta > 0$ un rayon. Supposons qu'il existe une constante positive κ telle que pour tout $y \in B(x;\Delta)$ le modèle m_f satisfait :*

$$\begin{aligned} \|\nabla^2 f(y) - \nabla m_f(y)\| &\leq \kappa\Delta, \\ \|\nabla f(y) - \nabla m_f(y)\| &\leq \kappa(\Delta)^2, \\ \|f(y) - m_f(y)\| &\leq \kappa(\Delta)^3. \end{aligned}$$

*Alors le modèle m_f est dit être un modèle κ-pleinement-quadratique (κ-*fully-quadratic *en anglais) de f sur $B(x;\Delta)$.*

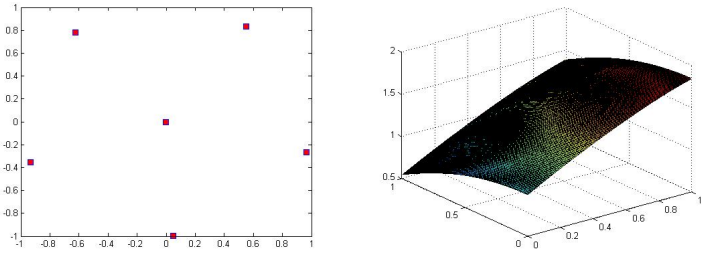
Afin de garantir que les modèles construits sont κ-pleinement-linéaires ou κ-pleinement-quadratiques, les auteurs de [43, chap. 2 à 4]) présentent la théorie du bon conditionnement (*well-poisedness* en anglais). Nous ne souhaitons pas entrer dans les détails complexes de la définition du bon conditionnement Λ, nous préférons ici donner un exemple de bon et de mauvais conditionnements :

Dans [43, chap. 6]) des algorithmes pour améliorer les ensembles d'échantillon de points afin de garantir que les modèles linéaires ou quadratiques construits par interpolation ou régression sont



Mauvais conditionnement

Figure extrait de [43]



Bon conditionnement

Figure extrait de [43]

pleinement-linéaire ou pleinement-quadratique sont présentés. Ces algorithmes peuvent nécessiter d'évaluer de nouveaux points pour définir l'échantillon de points utilisés pour construire le modèle.

2.3.2 Description de l'algorithme

Nous pouvons résumer les algorithmes DFTR de la manière suivante. Nous adoptons volontairement une présentation simplifiée afin de ne pas mélanger les détails techniques aux idées principales :

Un algorithme de région de confiance sans dérivées utilise des modèles calculés à partir d'un échantillon de points. L'algorithme est détaillé dans la figure 2.12.

Il y a plusieurs variantes de ces méthodes. Powell a beaucoup travaillé sur ces méthodes. C'est même lui qui a proposé en 1994 le premier algorithme de ce type, mais le formalisme de l'algorithme n'était pas celle proposée dans [43] quelques années plus tard.

Sous des hypothèses très strictes, dont certaines qui imposent que f soit continûment différentiable et que son gradient soit Lipschitz sur un certain ouvert, l'algorithme converge globalement vers un

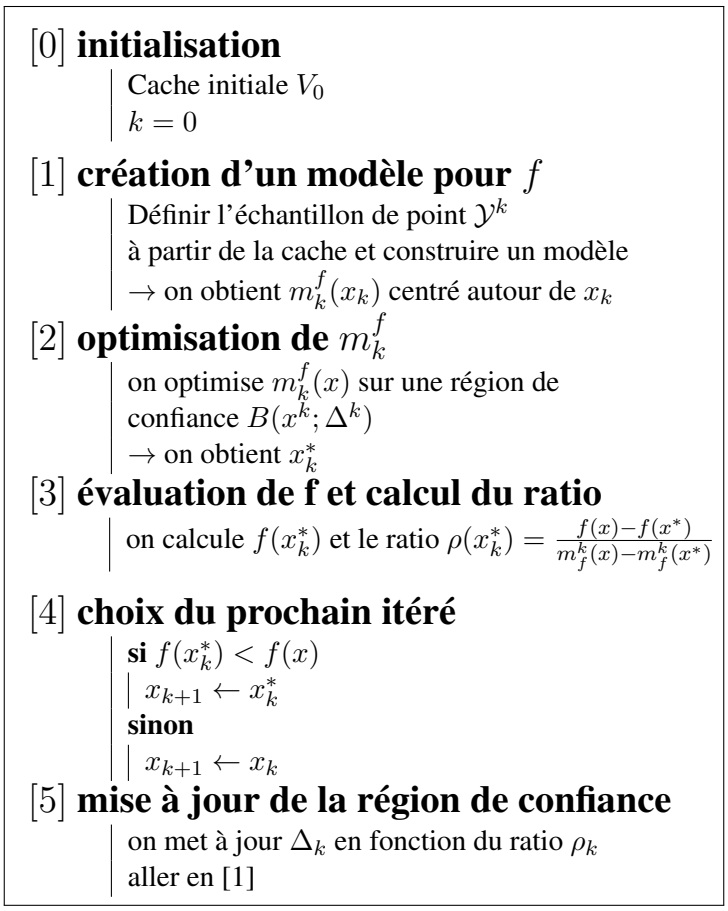


Figure 2.12 Résumé synthétique de l'algorithme DFTR sans contraintes.

point critique de premier ordre, c'est-à-dire un point tel que

$$\lim \nabla f(x_k) = 0.$$

L'algorithme devra aussi prendre soin de ne jamais diminuer la région de confiance lorsque les modèles ne sont pas certifiés de bonne qualité (par exemple κ-pleinement-linéaire ou κ-pleinement-quadratique pour un certain κ). En effet, si le ratio calculé est mauvais, la cause peut provenir non pas de la taille de la région de confiance mais d'un modèle inadéquat. Par ailleurs, lorsqu'un critère de convergence basé sur le gradient du modèle de f est satisfait, il faut vérifier que les modèles sont proches des vraies fonctions. Les définitions «κ-pleinement-linéaire» et «κ-pleinement-quadratique» donnent des bornes supérieures en fontion de Δ. Il faut alors vérifier que Δ n'est pas trop grand relativement à la norme du gradient du modèle pour que les modèles soient de bonnes approximations locales des vraies fonctions.

Une variante de l'algorithme peut être appliquée et offre des garanties de convergence vers un point

critique du second ordre (point en lequel le gradient tend vers 0 et la plus petite valeur propre de $\nabla^2 f(x_k)$ converge vers une valeur non négative). Ce résultat de convergence n'est obtenu que si, en plus des hypothèses précédentes, la Hessienne est Lipschitz localement.

Le point important à retenir sur les algorithmes de la famille DFTR est qu'ils n'utilisent pas les dérivées explicitement, mais supposent qu'elles existent. On construit des modèles qui sont de bonnes approximations locales des fonctions lorsqu'on s'approche des points critiques.

2.3.3 Traitement des contraintes

Le traitement des contraintes d'égalités et d'inégalités connues *a priori* est réalisé dans le sous-problème de l'algorithme de région de confiance sans dérivées lorsque la méthode de résolution du sous-problème le permet. C'est par exemple ce qui est fait dans [38], qui traite donc les contraintes a priori directement dans le sous-problème. Pour les contraintes plus complexes, comme des contraintes de *simulation* (contraire de contraintes *a priori*) relaxables, des modèles des contraintes sont construits à partir des mêmes échantillons de points que ceux utilisés pour construire le modèle de la fonction objectif. Le premier algorithme proposé est COBYLA, qui utilise des modèles linéaires. L'échantillon de points est composé de points aux sommets d'un simplexe. Les contraintes sont traitées en résolvant les sous-problèmes par des méthodes de contraintes actives.

Une fonction de pénalité avec la norme infinie est ainsi définie, et est utilisée pour calculer le ratio de la région de confiance. Récemment, Tröltzsch [102] adapte un algorithme séquentiel quadratique (*SQP* en anglais) dans un cadre d'algorithme de région de confiance sans dérivées, lui permettant de résoudre des problèmes avec contraintes d'égalités. Les auteurs de [104] proposent un algorithme DFTR combinant un Lagrangian augmenté avec une technique de filtre pour résoudre des problèmes spéciaux présentant une structure séparable. Sampaio et Toint adaptent dans [96] et [97] la méthode dite de l'entonnoir (*funnel*) de [55] pour des problèmes avec contraintes générales (égalités et inégalités, connues a priori ou connues de simulation). Un algorithme, proposé dans [26], définissant un critère fort de réalisabilité, nommé NOWPAC (pour Nonlinear Optimizer With Path-Augmented Contraints) permet de résoudre les contraintes à partir de points réalisables. Les modèles sont quadratiques.

Enfin, les auteurs de [10] et [50] proposent deux algorithmes DFTR et utilisent des méthodes à deux phases pour traiter les contraintes. Une première phase de restoration vise à obtenir un point réalisable. Enfin, des traitements spécifiques des contraintes linéaires sont proposées dans [58] et [92].

2.4 Lien entre algorithmes BBO et algorithmes DFO

On a déjà évoqué le lien ténu entre les problèmes BBO et les problèmes DFO. Il est à noter deux exemples d'algorithmes BBO implémentant des techniques lisses.

Tout d'abord, le logiciel HOPSPACK implémentant l'algorithme GSS utilise comme traitement des contraintes de simulation une adaptation du lagrangien Augmenté proposé dans [39].

Aussi, le logiciel NOMAD, qui implémente l'algorithme MADS, permet dans une première phase à chaque itération d'évaluer un nombre fini de points supplémentaires, autres que les points de sonde : c'est la phase de recherche globale, dite *Search* en anglais. Une proposition pour la phase de *Search* est proposée dans [40]. Un modèle quadratique est construit par régression à partir des points de la cache et un sous-problème est résolu afin d'obtenir un candidat à évaluer dans cette phase de *Search*.

CHAPITRE 3 ORGANISATION DE LA THÈSE

Cette thèse a pour objectif d'améliorer le traitement des contraintes dans les algorithmes DFO et BBO. Dans le chapitre 2 nous avons mis en évidence les différents types de problèmes, par rapport à la nature des fonctions en jeu mais aussi aux types de contraintes. Différentes techniques pour traiter les contraintes existent déjà, mais peu de traitements proposent d'exploiter les contraintes linéaires, et à l'inverse peu proposent de résoudre des problèmes de boîtes noires difficiles, avec des contraintes non approximables par des fonctions lisses, ou qui renvoit souvent des valeurs non exploitables. Les contraintes d'égalité en BBO demeurent une difficultée importante.

Dans un premier temps nous proposerons un traitement direct des contraintes d'égalité linéaire dans un algorithme BBO, mais qui peut aussi s'appliquer à un algorithme DFO. La méthode mise en œuvre permet de diminuer le degré de liberté du problème en travaillant dans le noyau affine défini par les contraintes d'égalités linéaires. Plusieurs convertisseurs permettant de reformuler le problème initial dans ce noyau sont proposées, et plusieurs stratégies les combinant sont analysées. Une méthode est retenue après comparaison numérique sur un ensemble de problèmes académiques. Notre méthode est globalement préférable à celle proposée par le logiciel HOPSPACK implémentant l'algorithme GSS avec le traitement des contraintes consistant, comme évoqué dans le chapitre précédent, à générer des directions parallèles aux cônes définissant les contraintes linéaires. Les travaux réalisés ont été publiés dans la revue Computational Optimization and Applications [24] et sont reportés dans le chapitre 4.

Dans un deuxième temps, nous nous intéresserons au traitement des contraintes connues par simulation dans les algorithmes DFO. Le deuxième objectif de cette thèse est de proposer une technique issue des algorithmes BBO dans un algorithme DFTR. Il s'agit d'adapter la barrière progressive, une méthodes BBO, initialement proposée pour l'algorithme MADS par [18], pour traiter dans un algorithme DFTR les contraintes d'inégalités générales. Dans le chapitre 2 nous avons mis en évidence les traitements des contraintes dans les algorithmes existants, et le peu de liens entre les algorithmes pour les problèmes BBO et ceux pour les problèmes DFO. Trop peu de relations existent entre les différentes familles d'algorithmes d'optimisation sans dérivées et proposer une méthode hybridant les techniques BBO et DFO est aussi une volonté de ce deuxième projet. L'algorithme obtenu présente l'avantage d'être compétitif avec COBYLA sur un ensemble de problèmes du type DFO et d'être compétitif avec NOMAD sur les problèmes BBO testés. Le chapitre 5 présente les travaux et les résultats obtenus dans ce deuxième projet qui ont été soumis à Computational Optimizations and Applications [14].

En troisième lieu, nous proposons un algorithme DFTR qui exploite simplement un algorithme de

Lagrangien augmenté pour traiter les problèmes d'égalité et d'inégalité de type simulation. L'algorithme obtenu est simple car il utilise une méthode de Lagrangian augmenté existant, et aussi car les règles de mise à jour de l'algorithme DFTR sont celles du cas sans contraintes adaptées à la fonction du Lagrangien augmenté obtenu à la fin de chaque résolution du sous-problème. L'algorithme permet un traitement des contraintes d'égalité apparemment meilleur que COBYLA sur l'ensemble de problèmes analytiques testés, comme le relate le chapitre 6. Ces travaux ont été soumis à Optimization Letters [25].

Finalement une discussion générale est présentée au chapitre 8.

CHAPITRE 4 ARTICLE 1 : LINEAR EQUALITIES IN BLACKBOX OPTIMIZATION

Recopié avec permission, C. Audet, S. Le Digabel et M. Peyrega, (2015), Linear Equalities in Blackbox Optimization. *Computational Optimization and Applications*, publié dans *Volume 61, Issue 1*, May 2015.

Abstract : The Mesh Adaptive Direct Search (MADS) algorithm is designed for blackbox optimization problems subject to general inequality constraints. Currently, MADS does not support equalities, neither in theory nor in practice. The present work proposes extensions to treat problems with linear equalities whose expression is known. The main idea consists in reformulating the optimization problem into an equivalent problem without equalities and possibly fewer optimization variables. Several such reformulations are proposed, involving orthogonal projections, QR or SVD decompositions, as well as simplex decompositions into basic and nonbasic variables. All of these strategies are studied within a unified convergence analysis, guaranteeing Clarke stationarity under mild conditions provided by a new result on the hypertangent cone. Numerical results on a subset of the CUTEst collection are reported.

4.1 Introduction

In some optimization problems, the objective function, as well as the functions defining the constraints, may be analytically unknown. They can instead be the result of an experiment or a computer simulation, and, as a consequence, they may be expensive to evaluate, be noisy, possess several local optima, and even return errors at a priori feasible points. Moreover, derivatives are unavailable and cannot be estimated, even when they exist, and therefore cannot be used for optimization. *Derivative-free optimization* (DFO) methods, and more precisely *direct-search* methods, are designed to handle these cases by considering only the function values. From aeronautics to chemical engineering going through medical engineering and hydrology, these algorithms have several applications in a wide range of fields. More details about these methods and their applications in numerous fields are exposed in the book [43] and in the recent survey [12].

The present work proposes a direct-search algorithm for blackbox optimization problems subject to general inequality constraints, lower and upper bounds, and linear equalities. Without any loss of generality, we consider only the linear equalities of the type $Ax = 0$, where A is *known a priori* and is a full row rank matrix. A simple linear translation can be applied to nullify a nonzero right-hand-side. We consider optimization problems of the following form :

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & F(x) \\ \text{subject to} \quad & C(x) \leq 0 \\ & Ax = 0 \\ & L \leq x \leq U, \end{aligned} \tag{4.1}$$

where $A \in \mathbb{R}^{m \times n_x}$ is full row rank matrix, $L, U \in (\mathbb{R} \cup \{-\infty\} \cup \{+\infty\})^{n_x}$ are bound vectors, possibly infinite, $F : \mathbb{R}^{n_x} \to \mathbb{R} \cup \{+\infty\}$ is a single-valued objective function, $C : \mathbb{R}^{n_x} \to (\mathbb{R} \cup \{+\infty\})^p$ is a vector of general inequality constraints, and n_x, m, $p \in \mathbb{N}$ are finite dimensions. Allowing the objective and inequality constraints to take infinite values is convenient in a minimization context for modeling situations in which the simulation failed to return a valid value.

An example of such a linear constrained derivative-free problem is exposed in [61]. Even if there is only one linear equality treated with a simple variable substitution, it suggests that other examples with more linear equalities could emerge from chemical engineering. Moreover, solving this kind of problem can have applications even to design more general DFO algorithms. Indeed, the authors of [33] design such algorithm for nonlinear constrained blackbox optimization. At each iteration, their method needs to solve linear equality constrained subproblems.

The objective of the present paper is to treat linear equalities with MADS by using various reformulations of the problem, seen as a wrapper on the original problem. In addition, we prove a new result on the hypertangent cone, which allows to extend the MADS convergence theory.

The document is organized as follows. Section 4.2 reviews literature about equalities in DFO. Section 4.3 presents the basic framework to transform Problem (4.1) into a problem on which MADS can be applied, as well as a proof that, under mild conditions, this framework inherits the MADS convergence properties. Section 4.4 then introduces four classes of transformations implementing the basic framework, and Section 4.5 illustrates the efficiency of these transformations, as well as some hybrid strategies. We conclude with a discussion and future work in Section 5.5.

4.2 Handling linear constraints in DFO

Derivative-free algorithms may be partitioned into direct-search and model-based methods. At each iteration, direct-search methods evaluate the functions defining the problem (the blackbox) at a finite number of points and take decision for the next step based on these values. In model-based methods, models are constructed to approximate the blackbox and are then optimized to generate candidates where to evaluate the blackbox.

The first direct-search algorithm considering linearly constrained problems is proposed by May [80]. It extends Mifflin's derivative-free unconstrained minimization algorithm [81], a hybrid derivative-free method combining direct-search ideas with model-based tools. May's main contribution is to use positive generators of specific cones which are approximations of the tangent cones of the active constraints. He proves both global convergence and superlinear local convergence under assumptions including continuous differentiability of F. Later, Lewis and Torczon [71] proposed the Generalized Pattern Search (GPS) algorithm [100] to treat problems subject to linear inequalities. Under assumptions including continuous differentiability of F and rationality of the constraints matrix, they show convergence to a KKT point, even in the presence of degeneracy. Other improvements to deal with degeneracy are proposed in [6], and similar ideas are given in [75] where the unconstrained GPS algorithm is adapted with directions in the nullspace of the matrix A. Coope and Price extended these ideas to a grid-based algorithm [45] by aligning the positive basis with the active set of linear constraints [93]. These extensions were adapted to the Generating Set Search (GSS) method [57, 66, 69, 73], which allows a broader selection of search directions at the expense of imposing a minimal decrease condition in order to accept a new incumbent solution. The positive basis for GSS at each iteration is chosen in the nullspace of active equality constraints. This approach reduces the dimension of the problem, because the search directions at each iteration are contained in a subspace. References [65] and [74] present an algorithm for differentiable nonlinear problems with nonlinear equality and inequality constraints. An augmented Lagrangian method is adapted in the GSS framework, which provides a special treatment for linear constraints. This method for differentiable equalities and inequalities was first proposed in a trust-region context in [39]. These ideas are implemented in the HOPSPACK software package [87].

The subject of equality constraints in derivative-free optimization was previously addressed. The approach suggested in the current paper is quite close to the one of [49], where the problem to be solved includes general equality constraints, implicitly treated as defining a manifold. The author of [49] explicitly mentions the usefulness of the proposed procedure to allow the extension of MADS to deal with equality constraints and discusses the benefits of reducing the problem dimension. The current work is focused on linear equalities, rather than general equality constraints, which simplifies the analysis and presentation. In [79], Martinez and Sobral propose a direct-search algorithm which can treat equality constraints by relaxing them into two inequalities, resulting in a narrow domain. Finally, [98] extends MADS to handle sphere equality constraints.

Derivative-free trust-region methods are a class of model-based algorithms, using regression or interpolation. The theory considers no constraints but it can be easily adapted to bound and linearly constrained problems [43]. Moreover, the equalities are used to reduce the degrees of freedom of the problem [41]. In the LINCOA package [90, 91], Powell proposes an implementation of a derivative-free trust-region algorithm considering linear inequality contraints by using an active set method.

Finally, in [96] Sampaio and Toint propose an algorithm treating general equality constraints by using a trust-funnel method [55].

The Mesh Adaptive Direct Search (MADS) algorithm [17] is a direct-search method generalizing GPS and designed for bound-constrained blackbox problems. Its convergence analysis is based on the Clarke calculus [37] for nonsmooth functions. Inequalities including linear constraints are treated with the progressive barrier technique of [18], and equality constraints are not currently supported. The idea exposed in Section 4.3 is to treat linear equalities by designing a wrapper and a converter to transform the initial Problem (4.1) into a problem on which MADS can be applied.

4.3 Reformulations without linear equalities

The approach proposed in the present work consists of reformulating Problem (4.1) by eliminating the linear constraints $Ax = 0$, and then by applying the MADS algorithm to the reformulation. The following notation is used throughout the paper. Let $S = \{x \in \mathbb{R}^{n_x} \ : \ Ax = 0\}$ denote the nullspace of the matrix A, and $\Omega_x = \{x \in S \ : \ C(x) \leq 0, \ L \leq x \leq U\}$ the feasible set to Problem (4.1).

A straightforward suggestion is to transform the linear equality $Ax = 0$ into two inequalities, and to include them to the inequality constraints $C(x) \leq 0$, as suggested in the LINCOA package [90, 91]. However, this method is unsuitable in our context for both theoretical and practical issues. First, existing convergence analysis of MADS would be limited to its most basic result. The convergence analysis of MADS leading to Clarke stationarity requires that the hypertangent cone to the feasible set Ω_x is non-empty at some limit point (Section 4.3.3 of the present document presents the definition of the hypertangent cone relative to a subspace). However, in the presence of linear equalities, the hypertangent cone is necessarily empty for every $x \in \mathbb{R}^{n_x}$. This is true because the hypertangent cone is an open set [37] and therefore it is either empty or its dimension equals n_x. Second, almost all evaluations will occur outside of the linear subspace S. The extreme barrier [17] will reject these points, or the progressive barrier [18] will invest most of its effort in reaching feasibility rather than improving the objective function value. This explains that splitting an equality into two inequalities fails in practice, as observed in [33]. In addition, with such reformulations, MADS fails to find a feasible point for all the problems tested in Section 4.5.

The ideas introduced for GPS and GSS as described in the previous section could also be implemented in MADS. Instead of choosing orthogonal positive bases in the entire space $\mathbb{R}^{n_x}$ to generate the mesh in MADS, it is possible to choose a positive basis of S, and to complete it by directions in $\mathbb{R}^{n_x} \setminus S$ in order to obtain a positive basis of $\mathbb{R}^{n_x}$. These directions would be pruned by the algorithm because they generate points outside of S. Some of the strategies presented in Section 4.4

can be seen as particular instantiations of this approach. However, we prefer to view them from a different perspective.

4.3.1 Inequality constrained reformulation

In order to reformulate Problem (4.1) into an inequality constrained optimization problem over a different set of variables in $\mathbb{R}^{n_z}$, we introduce the following definition.

Definition 4.3.1 *A converter φ is a surjective linear application from $\mathbb{R}^{n_z}$ to $S \subset \mathbb{R}^{n_x}$, for some $n_z \in \mathbb{N}$. For any $x \in S$, there exists an element $z \in \mathbb{R}^{n_z}$ such that $x = \varphi(z)$.*

By definition, a converter φ is continuous, and any $z \in \mathbb{R}^{n_z}$ is mapped to an $x = \varphi(z) \in \mathbb{R}^{n_x}$ that satisfies the linear equalities $Ax = 0$. Since $dim\ S$ is equal to $n_x - m$, it follows that $n_z \geq n_x - m$. Furthermore, n_z equals $n_x - m$ if and only if the converter is bijective.

A converter φ is used to perform a change of variables, and to reformulate Problem (4.1) as follows :

$$\begin{array}{ll} \min\limits_{z \in \mathbb{R}^{n_z}} & f(z) \\ \text{subject to} & c(z) \leq 0 \\ & L \leq \varphi(z) \leq U\ . \end{array} \tag{4.2}$$

where $f : \mathbb{R}^{n_z} \to (\mathbb{R} \cup \{+\infty\})$ and $c : \mathbb{R}^{n_z} \to (\mathbb{R} \cup \{\pm\infty\})^p$ are defined by :

$$f(z) = F(\varphi(z)) \quad \text{and} \quad c(z) = C(\varphi(z))\ .$$

Let $\Omega_z = \{z \in \mathbb{R}^{n_z} : c(z) \leq 0 \text{ and } L \leq \varphi(z) \leq U\}$ denote the set of feasible solutions for the reformulated Problem (4.2). The following proposition details the connection between the sets of feasible solutions for both sets of variables x and z.

Proposition 4.3.2 *The image of Ω_z by the converter φ is equal to Ω_x, and the preimage of Ω_x by converter φ is equal to Ω_z :*

$$\Omega_x = \varphi(\Omega_z) \quad \textit{and} \quad \Omega_z = \varphi^{-1}(\Omega_x) := \{z \in \mathbb{R}^{n_z} \ :\ \varphi(z) \in \Omega_x\}.$$

PROOF. By construction we have $\varphi(\Omega_z) \subset \Omega_x = \{x \in S :\ C(x) \leq 0,\ L \leq x \leq U\}$. In addition, as φ is surjective, for any $x \in \Omega_x \subset S$, there exists some $z \in \mathbb{R}^{n_z}$ such that $\varphi(z) = x$. Since x belongs to Ω_x, it follows that $C(\varphi(z)) \leq 0$ and $L \leq \varphi(z) \leq U$. Therefore, $z \in \Omega_z$ and $x \in \varphi(\Omega_z)$. This implies that $\Omega_x \subset \varphi(\Omega_z)$.

Conversely, as Ω_x is equal to $\varphi(\Omega_z)$ and by the definition of the preimage, we have $\Omega_z \subset \varphi^{-1}(\Omega_x)$. Moreover, if $z \in \varphi^{-1}(\Omega_x)$, then $\varphi(z) \in \Omega_x$ and $z \in \Omega_z$. This implies that $\varphi^{-1}(\Omega_x) \subset \Omega_z$. □

The converter is a generalization of variable elimination techniques used for linear constraints, as described for example in Chapter 21 and 22 of [34] and in Chapter 15 of [84]. The classes of converters defined in Section 4.4 use matrix decomposition techniques presented in [34] and [84], in other optimization contexts.

4.3.2 Applying MADS on a reformulation

The converter is used to construct a wrapper around the original optimization problem so that the MADS algorithm is applied to the reformulated one. Figure 4.1 illustrates the application of the MADS algorithm to Problem (4.2). MADS proposes trial points $z \in \mathbb{R}^{n_z}$, which are converted into $x = \varphi(z)$ belonging to the nullspace S. If x is within the bounds of the original optimization problem, then the blackbox simulation is launched to evaluate $F(x)$ and $C(x)$. Otherwise, the cost of the simulation is avoided and $F(x)$ and $C(x)$ are arbitrarily set to an infinite value. In both cases, the outputs are assigned to $f(z)$ and $c(z)$, and returned to the MADS algorithm.

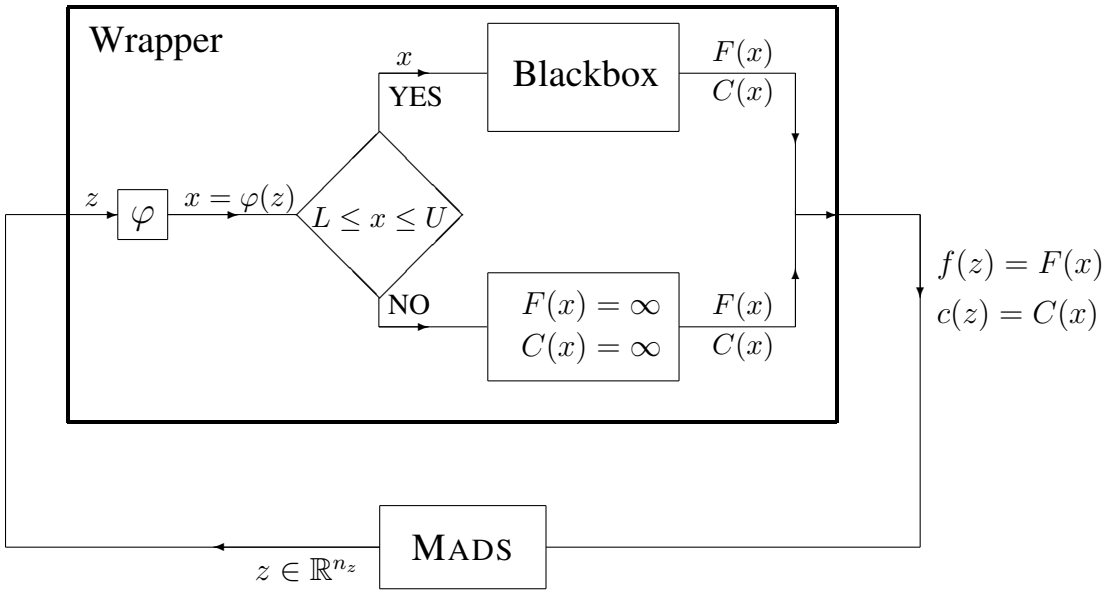


Figure 4.1 The converter φ allows the construction of a wrapper around the original blackbox.

The MADS algorithm can be applied to Problem (4.2) and the constraints can be partitioned into two groups. The constraint functions $c(z)$ are evaluated by launching the blackbox simulation; the constraints $L \leq \varphi(z) \leq U$ are checked *a priori*, before executing the blackbox. When these constraints are not satisfied for a given $z \in \mathbb{R}^{n_z}$, the cost of launching the blackbox is avoided.

A preprocessing phase can be executed to delimit more precisely the domain Ω_z. For each $i \in \{1, 2, \ldots, n_z\}$, solving the following linear programs yields valid lower and upper bounds $\ell, u \in (\mathbb{R} \cup \{-\infty\} \cup \{+\infty\})^{n_z}$ on z :

$$\ell_i = \min \{z_i \ : \ L \leq \varphi(z) \leq U, z \in \mathbb{R}^{n_z}\},$$
$$u_i = \max \{z_i \ : \ L \leq \varphi(z) \leq U, z \in \mathbb{R}^{n_z}\}.$$

Thus, the problem that MADS considers in practice, equivalent to Problems (4.1) and (4.2), is the following :

$$\begin{array}{rl} \min\limits_{z \in \mathbb{R}^{n_z}} & f(z) \\ \text{subject to} & c(z) \leq 0 \\ & L \leq \varphi(z) \leq U \\ & \ell \leq z \leq u \, . \end{array} \tag{4.3}$$

The feasible set for this problem is Ω_z, as for Problem (4.2). The difference is that the bounds ℓ and u may be used by MADS to scale the variables.

4.3.3 Convergence Analysis

The fundamental convergence result [17] of MADS studies some specific accumulation points of the sequence of trial points in $\mathbb{R}^{n_z}$. In the proposed approach, the original problem is formulated in R^{n_x}, but the algorithm is deployed in $\mathbb{R}^{n_z}$. Our convergence analysis consists in transposing the theoretical results from $\mathbb{R}^{n_z}$ to the nullspace $S \subset \mathbb{R}^{n_x}$, which contains the entire sequence of trial points.

We use superscripts to distinguish vector spaces. For example, if E is a normed vector space like $\mathbb{R}^{n_z}, \mathbb{R}^{n_x}$, or S, we will denote the open ball of radius $\varepsilon > 0$ centred at $x \in E$ by :

$$B_\varepsilon^E(x) := \{y \in E : ||y - x|| < \varepsilon\} .$$

The convergence analysis relies on the following definition of the Clarke derivative taken from [62] and adapted with our notations.

Definition 4.3.3 *Let Ω be a nonempty subset of a normed vector space E, $g : \Omega \longrightarrow \mathbb{R}$ be Lipschitz*

near a given $x \in \Omega$, and let $v \in E$. The Clarke generalized derivative at x in the direction v is :

$$g^\circ(x;v) \;:=\; \limsup_{\substack{y \to x,\ y \in \Omega \\ t \downarrow 0,\ y+tv \in \Omega}} \frac{g(y+tv)-g(y)}{t}\,. \tag{4.4}$$

An important difference with previous analyses of MADS is that E may be a strict subset of a greater space. For example, in our context, E corresponds to the nullspace S, strictly contained in $\mathbb{R}^{n_x}$, and $\mathbb{R}^{n_x}$ is the native space of the original optimization problem.

The next definition describes the hypertangent cone $T_\Omega^H(x)$ to a subset $\Omega \subseteq E$ at x, where E is a normed vector space, as given by Clarke [37].

Definition 4.3.4 *Let Ω be a nonempty subset of a normed vector space E. A vector $v \in E$ is said to be hypertangent to the set Ω at the point $x \in \Omega$ if there exists a scalar $\varepsilon > 0$ such that :*

$$y + tw \in \Omega \quad \textit{for all } y \in B_\varepsilon^E(x) \cap \Omega,\ \ w \in B_\varepsilon^E(v) \ \textit{ and } \ 0 < t < \varepsilon\,. \tag{4.5}$$

The set of hypertangent vectors to Ω at x is called the hypertangent cone to Ω at x and is denoted by $T_\Omega^H(x)$.

A property of the hypertangent cone is that it is an open cone in the vector space E. The convergence analysis below relies on the assumption made for the MADS convergence analysis (see [17] for more details). The following theorem asserts that hypertangent cone mapped by the converter φ coincides with the hypertangent cone in the nullspace S at the point mapped by φ.

Theorem 4.3.5 *For every $z \in \Omega_z$, the hypertangent cone to Ω_x at $x = \varphi(z)$ in $\mathbb{R}^{n_z}$ is equal to the image by φ of the hypertangent cone to Ω_z at z in S. In other words,*

$$T_{\Omega_x}^H(\varphi(z)) = \varphi(T_{\Omega_z}^H(z))\,.$$

PROOF. The equality is shown by double inclusion. Both inclusions use the linearity of the converter φ and Proposition 4.3.2. The first inclusion is based on the continuity of φ while the second requires the open mapping theorem [95]. The cases $T_{\Omega_x}^H(\varphi(z)) = \emptyset$ and $T_{\Omega_z}^H(z) = \emptyset$ are trivial. In the following, let z be an element of the nonempty set Ω_z.

First inclusion proof. Let $v \in T_{\Omega_x}^H(\varphi(z)) \subseteq S$ be an hypertangent vector to Ω_x, and let $d \in \mathbb{R}^{n_z}$ be such that $\varphi(d) = v$. We show that d is hypertangent to Ω_z at z. By Definition 4.3.4, choose $\varepsilon > 0$

such that :

$$y + tw \in \Omega_x \quad \text{for all } y \in B_\varepsilon^S(\varphi(z)) \cap \Omega_x,\ w \in B_\varepsilon^S(\varphi(d)) \ \text{ and } \ 0 < t < \varepsilon\ . \tag{4.6}$$

Continuity of φ and Proposition 4.3.2 allow to select $\varepsilon_1 > 0$ and $\epsilon_2 > 0$ sufficiently small so that :

$$\varphi(B_{\varepsilon_1}^{\mathbb{R}^{n_z}}(z) \cap \Omega_z) \subset \left(B_\varepsilon^S(\varphi(z)) \cap \Omega_x\right) \quad \text{and} \quad \varphi(B_{\varepsilon_2}^{\mathbb{R}^{n_z}}(d)) \subset B_\varepsilon^S(\varphi(d)).$$

Let define $\varepsilon_{\min} := \min\{\varepsilon_1, \varepsilon_2, \varepsilon\}$, $r \in \left(B_{\varepsilon_{min}}^{\mathbb{R}^{n_z}}(z) \cap \Omega_z\right)$, $s \in B_{\varepsilon_{min}}^{\mathbb{R}^{n_z}}(d)$, and $0 < t < \varepsilon_{\min}$. It follows that $y := \varphi(r) \in \left(B_\varepsilon^S(\varphi(z)) \cap \Omega_x\right)$ and $w := \varphi(s) \in B_\varepsilon^S(\varphi(d))$. Thus, Assertion (4.6) and linearity of φ ensure that :

$$\varphi(r + ts) = \varphi(r) + t\varphi(s) = y + tw \in \Omega_x\ .$$

Finally, since $\Omega_z = \varphi^{-1}(\Omega_x)$, it follows that $r + ts \in \Omega_z$. Definition 4.3.4 is satisfied with r, s, t and $\varepsilon_{\min}$ and therefore $d \in T_{\Omega_z}^H(z)$ implies that $v = \varphi(d) \in \varphi(T_{\Omega_z}^H(z))$.

Second inclusion proof. Let $d \in T_{\Omega_z}^H(z)$ be an hypertangent vector to Ω_z at z. We show that $\varphi(d) \in \varphi(T_{\Omega_z}^H(z))$ is hypertangent to Ω_x at $\varphi(z)$. By Definition 4.3.4, choose $\varepsilon > 0$ such that :

$$r + ts \in \Omega_z \text{ for all } \ r \in B_\varepsilon^{\mathbb{R}^{n_z}}(z) \cap \Omega_z,\, s \in B_\varepsilon^{\mathbb{R}^{n_z}}(d) \text{ and } 0 < t < \varepsilon.$$

The open mapping theorem [95] ensures that there exist $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ such that :

$$B_{\varepsilon_1}^S(\varphi(z)) \subset \varphi(B_\varepsilon^{\mathbb{R}^{n_z}}(z)) \quad \text{and} \quad B_{\varepsilon_2}^S(\varphi(d)) \subset \varphi(B_\varepsilon^{\mathbb{R}^{n_z}}(d)).$$

Define $\varepsilon_{\min} := \min\{\varepsilon_1, \varepsilon_2, \varepsilon\}$, and let

$$y \in \left(B_{\varepsilon_{min}}^S(\varphi(z)) \cap \Omega_x\right), w \in B_{\varepsilon_{min}}^S(\varphi(d)) \ \text{ and } \ 0 < t < \varepsilon_{\min}\ .$$

By the choice of $\varepsilon_{\min}$, it follows that y belongs to both sets $\varphi(B_\varepsilon^{\mathbb{R}^{n_z}}(z))$ and Ω_x. Consequently, there exists an $r \in B_\varepsilon^{\mathbb{R}^{n_z}}(z)$ such that $y = \varphi(r)$, which also belongs to $\Omega_z = \varphi^{-1}(\Omega_x)$ since $\varphi(r) \in \Omega_x$. Let $s \in B_\varepsilon^{\mathbb{R}^{n_z}}(d)$ be such that $w = \varphi(s)$. Applying the converter φ yields :

$$y + tw = \varphi(r) + t\varphi(s) = \varphi(r + ts) \in \Omega_x$$

since $r+ts \in \Omega_z$. Definition 4.3.4 is satisfied with y, w, t and $\varepsilon_{\min}$, and therefore $\varphi(d) \in T_{\Omega_x}^H(\varphi(z))$.

□

In our algorithmic framework, we apply the MADS algorithm to Problem (4.3) which is an equivalent reformulation of Problem (4.1). We use the standard assumptions [5, 17] that the sequence of iterates produced by the algorithm belongs to a bounded set, and that the set of normalized polling directions is asymptotically dense in the unit sphere. The MADS convergence analysis [17] gives conditions ensuring the existence of a refined point, i.e., a cluster point of the sequence of trial points at which $f^\circ(z^*; v) \geq 0$ for every hypertangent direction $v \in T^H_{\Omega_z}(z^*)$, provided that f is locally Lipschitz near z^*. However, this result holds on the reformulated problem, and is not stated using the notations of the original equality constrained problem. The following theorem fills the gap by stating the main convergence result for Problem (4.3).

Theorem 4.3.6 *Let x^* be the image of a refined point z^* produced by the application of* MADS *on Problem* (4.3) *: $x^* = \varphi(z^*)$. If F is locally Lipschitz near x^*, then :*

$$F^\circ(x^*; v) \geq 0 \quad \textit{for all } \; v \in T^H_{\Omega_x}(x^*)\,.$$

PROOF. Let z^* be a refined point produced by the application of MADS to Problem (4.3) and set $x^* := \varphi(z^*)$ be the corresponding point in the original space of variables.

Let $v \in T^H_{\Omega_x}(x^*) = T^H_{\Omega_x}(\varphi(z^*))$ be an hypertangent direction at x^*. By Proposition 4.3.5, let $d \in T^H_{\Omega_z}(z^*)$ be such that $\varphi(d) = v$.

If F is locally Lipschitz near x^*, and since φ is a linear application, then the definition of $f(z) = F(\varphi(z))$ ensures that f is locally Lipschitz near z^*. The MADS convergence result holds : $f^\circ(z^*; d) \geq 0$. By Definition (4.4), let $t_k \to 0$ be a sequence in $\mathbb{R}$, and $r_k \to z^*$ and $s_k \to d$ be two sequences such that both r_k and $r_k + t_k s_k$ belong to Ω_z and :

$$f(r_k + t_k s_k) - f(r_k) \geq 0 \quad \text{for every } \; k \in \mathbb{N}.$$

The converted sequence $\{y_k\} := \{\varphi(r_k)\}$ and $\{w_k\} := \{\varphi(s_k)\}$ converge respectively to x^* and v, and satisfy for every $k \in \mathbb{N}$:

$$\begin{aligned} f(r_k + t_k s_k) - f(r_k) \geq 0 \quad &\Longleftrightarrow \quad F(\varphi(r_k + t_k s_k)) - F(\varphi(r_k)) \geq 0 \\ &\Longleftrightarrow \quad F(y_k + t_k w_k) - F(y_k) \geq 0\,. \end{aligned}$$

which shows that $F^\circ(x^*; v) \geq 0$.

□

Corollaries of this theorem can be developed as in [17] by analyzing smoother objective functions, or by imposing more conditions on the domain Ω_z. For example, by imposing strict differentiability

of F near x^* and by imposing that Ω_x is regular and that the hypertangent cone is nonempty, then one can show that x^* is a contingent KKT stationary point of F over Ω_x.

4.4 Different classes of transformations

A converter is a surjective linear application. In this section, we present four different converters based on an orthogonal projection, SVD and QR decompositions, and a simplex-type decomposition that partitions the variables into basic and non-basic variables, as exposed in [84]. For each converter, we describe the φ function and show that it maps any vector onto the nullspace S.

4.4.1 Orthogonal projection

Define $\varphi_P : \mathbb{R}^{n_x} \to S$ as the orthogonal projection of the matrix A into the nullspace S. For each $z \in \mathbb{R}^{n_x}$, define :

$$\varphi_P(z) := (I - A^+A)z \in S \subset \mathbb{R}^{n_x}$$

where $A^+ = A^T(AA^T)^{-1}$ is the pseudoinverse of A. The inverse of AA^T exists because A is of full row rank. If $x = \varphi_P(z)$, then :

$$Ax \;=\; A(I - A^+A)z \;=\; (A - AA^T(AA^T)^{-1}A)z \;=\; 0$$

which confirms that $x \in S$. For this converter, $n_z = n_x$.

4.4.2 QR decomposition

The QR decomposition of the matrix $A^T \in \mathbb{R}^{n_x \times m}$ is $A^T = QR$, where $Q \in \mathbb{R}^{n_x \times n_x}$ is an orthogonal matrix and $R \in \mathbb{R}^{n_x \times m}$ is an upper triangular matrix. Furthermore,

$$A^T = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

where $Q_1 \in \mathbb{R}^{n_x \times m}$ and $Q_2 \in \mathbb{R}^{n_x \times (n_x - m)}$ are composed of orthonormal vectors, and $R_1 \in \mathbb{R}^{m \times m}$ is an upper triangular square matrix. Finally, 0 corresponds to the null matrix in $R^{(n_x - m) \times m}$. For all $z \in \mathbb{R}^{n_x - m}$, the converter $\varphi_{QR} : \mathbb{R}^{n_x - m} \to S$ is defined as :

$$\varphi_{QR}(z) := Q_2 z \; .$$

If $x = \varphi_{QR}(z)$, and since Q is an orthogonal matrix, then

$$Ax = \begin{pmatrix} R_1^T & 0^T \end{pmatrix} \begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix} Q_2 z = \begin{pmatrix} R_1^T & 0^T \end{pmatrix} \begin{pmatrix} 0 \\ I \end{pmatrix} z = 0$$

which shows that $x \in S$. For this converter, $n_z = n_x - m$.

4.4.3 SVD decomposition

Unlike the diagonalization which cannot be applied to every matrix, Singular Value Decomposition (SVD) is always possible. The full row rank matrix A of Problem (4.1) can be decomposed in $A = W\Sigma V^T$ where $W \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n_x \times n_x}$ are unitary matrices, and Σ can be written as :

$$\Sigma = \begin{pmatrix} \sigma_1 & \cdots & 0 & 0 \cdots 0 \\ \vdots & \ddots & \vdots & 0 \cdots 0 \\ 0 & \cdots & \sigma_m & 0 \cdots 0 \end{pmatrix} \in \mathbb{R}^{m \times n_x}$$

for some positive scalars σ_i, $i \in \{1, 2, \ldots, m\}$. Since W and V are unitary matrices, then $W^{-1} = W^T$ and $V^{-1} = V^T$. For all $z \in \mathbb{R}^{n_x - m}$, the converter $\varphi_{SVD} : \mathbb{R}^{n_x - m} \to S$ is defined as :

$$\varphi_{SVD}(z) := V \begin{pmatrix} 0_m \\ z \end{pmatrix}$$

where 0_m is the null vector in $\mathbb{R}^m$. If $x = \varphi_{SVD}(z)$, then :

$$Ax = W\Sigma V^T V \begin{pmatrix} 0_m \\ z \end{pmatrix} = W\Sigma \begin{pmatrix} 0_m \\ z \end{pmatrix} = 0$$

and therefore $x \in S$. For this converter, $n_z = n_x - m$.

4.4.4 BN decomposition

This fourth converter uses the simplex-type decomposition into basic and nonbasic variables. It reduces Problem (4.1) to one with dimension $n_x - m$. The full row rank matrix A has more columns than rows. Let I_B and I_N form a partition of the columns of A such that $B = (A_i)_{i \in I_B}$ is a nonsingular $m \times m$ matrix, and $N = (A_i)_{i \in I_N}$ is a $m \times (n_x - m)$ matrix. The vector x is partitioned in the same way :

$$A = [B\ N] \qquad \text{and} \qquad x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$$

where $x_B = \{x_i : i \in I_B\}$ is of dimension m and $x_N = \{x_i : i \in I_N\}$ of dimension $n_x - m$. For any nonbasic variable $x_N \in \mathbb{R}^{n_x - m}$, setting $x_B = -B^{-1}Nx_N$ implies that x satisfies the linear equalities. For all $z \in \mathbb{R}^{n_x - m}$, the converter $\varphi_{BN} : \mathbb{R}^{n_x - m} \to S$ is defined as :

$$\varphi_{BN}(z) := \begin{pmatrix} -B^{-1}Nz \\ z \end{pmatrix},$$

and if $x = \varphi_{BN}(z)$, then

$$Ax = [B\ N] \begin{pmatrix} -B^{-1}Nz \\ z \end{pmatrix} = -BB^{-1}Nz + Nz = 0$$

which confirms that $x \in S$. The optimization problem on which MADS is applied has $n_z = n_x - m$ variables and can be written as follows :

$$\begin{array}{ll} \min\limits_{z \in \mathbb{R}^{n_z}} & f_{BN}(z) \\ \text{subject to} & c_{BN}(z) \leq 0 \\ & L \leq \varphi_{BN}(z) \leq U \end{array} \tag{4.7}$$

where $f_{BN}(z) = F(\varphi_{BN}(z))$ and $c_{BN}(z) = C(\varphi_{BN}(z))$. Note that the choice of matrices (B, N) is not unique.

4.4.5 Comments on the converters

The converters presented above reduce the dimension of the space of variables on which MADS is applied from n_x to $n_z = n_x - m$, where m is the number of linear equalities, except for the orthogonal projection technique of Section 4.4.1 for which the dimension remains n_x. The projection is also the only transformation that is not bijective. For the first three converters (P, QR and SVD), the bounds $L \leq x \leq U$ are translated into linear inequalities, which are treated as *a priori* constraints, as shown in Figure 4.1. The BN decomposition partitions the variables as basic and nonbasic variables, and optimizes over the nonbasic ones. Therefore, their bounds are simply $L_i \leq z_i \leq U_i$, for $i \in \{1, 2, \ldots, n_z\}$. Moreover, these first three converters are uniquely determined. However, there are exponentially many ways to partition the variables into basic and nonbasic ones. A practical way to identify a partition is described in Section 4.5.2 of the numerical experiments.

4.5 Implementation and numerical results

This section presents the implementation and numerical experiments of the strategies for handling the linear equalities. First, the set of test problems is presented. Then, the four converters are compared. Finally, strategies combining different converters are proposed. All initial points considered in this section are publicly available at `https://www.gerad.ca/Charles.Audet/publications.html`.

4.5.1 Numerical testbed

We consider 10 problems from the CUTEst collection [54] including 6 from the Hock & Schittkowski collection [60] and 4 others tested in [66, 73, 74, 69]. The degree of freedom n ranges from 2 to 40, which is representative of the degree of freedom that DFO algorithms can usually solve. Most problems have bounds, and two of them have linear inequality constraints. Linear inequalities are treated a priori, similarly to the bounds in Figure 4.1. One problem has quadratic inequality constraints. Names, dimensions and information on these analytical problems are summarized in Table 4.1.

Table 4.1 Description of the 10 CUTEst analytical problems.

Name	n_x	m	n	lower bounds	upper bounds	linear ineq.	quad. ineq.
HS48	5	2	3	0	0	0	0
HS49	5	2	3	0	0	0	0
HS51	5	3	2	0	0	0	0
HS53	5	3	2	5	5	0	0
HS112	10	3	7	10	0	0	0
HS119	16	8	8	16	16	0	0
DALLASS	46	30 [1]	16	46	46	0	0
LOADBAL	31	11	20	20	11	20	0
AVION2	49	15	34	49	49	0	0
PRODPl0	60	20	40	60	0	5	4 [2]

For each problem, 100 different initial feasible points are randomly generated, yielding a total of 1,000 instances. Each instance is treated with a maximal budget of 100(n+1) objective function evaluations where $n = n_x - m$ is the degree of freedom. The value 100(n+1) is typical of the number of evaluations used for plotting data profiles in DFO tests, as in [22].

In the next subsections, data profiles [83] are plotted to analyse the results. These graphs compare a set of algorithms on a set of instances for a relative tolerance $\alpha \in [0; 1]$. Each algorithm corresponds to a plot where each couple (x, y) indicates the proportion y of problems solved within the relative tolerance α after x groups of $n + 1$ evaluations. The relative tolerance α is used to calculate the threshold below which an algorithm is considered to solve a specific instance successfully. This threshold is defined as the best value obtained for this instance by any algorithm tested, with an added allowance of α multiplied by the improvement between the initial value and that best value. The value of α used in this section is set to 1%.

The NOMAD [2, 67] (version 3.6.0) and HOPSPACK [87] (version 2.0.2) software packages are used with their default settings, except for the following : the use of models in NOMAD is disabled, and the tolerance for the stopping criteria in HOPSPACK is set to 1E-13, which is comparable to the equivalent NOMAD parameter. In HOPSPACK, the GSS algorithm named citizen 1 is considered.

1. The original matrix of constraints has 31 rows but a rank of 30.
2. The formulation contains 4 quadratic constraints that can be formulated as linear constraints. We consider the quadratic formulation to execute our algorithms.

4.5.2 BN analysis and implementation

There can be up to $\binom{n_x}{m}$ different partitions (I_B, I_N) of matrix A, and every choice is not equivalent in practice. To illustrate this observation, we consider the HS119 problem with the starting point suggested in [60] using all 12,464 feasible partition schemes. Nocedal and Wright [84] suggest to partition the matrix in a way so that B is well-conditioned. Figure 4.2 plots the final objective function values produced by these executions after 900 objective function evaluations versus the condition number of the nonsingular matrix B, using a logarithmic scale. Most partitions of the matrix A lead to final objective function values far away from the optimal. This suggests that arbitrarily choosing a partition may lead to unsatisfactory solutions.

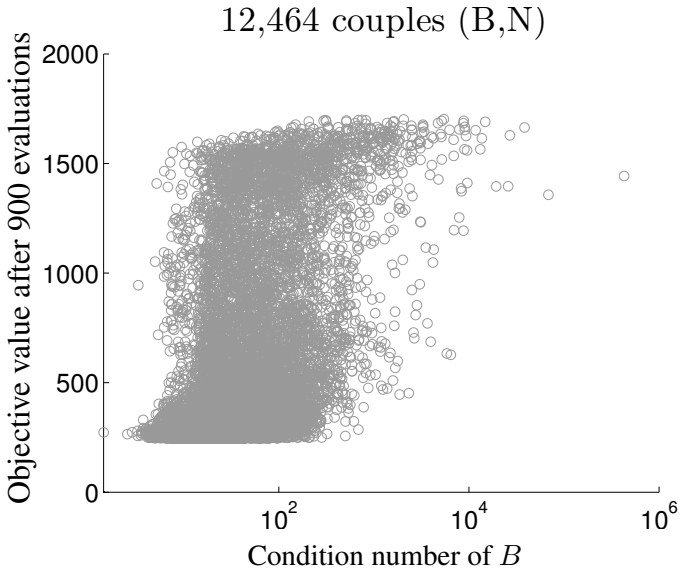


Figure 4.2 Execution of the BN algorithm on the HS119 problem with 900 evaluations.

The figure also reveals a trend that the quality of the solution increases when the condition number decreases. If the condition number exceeds 10^3, then the algorithm always fails to approach the optimal solution within the available budget of evaluations. However, even when the condition number of B is small, the final solution may be far from the optimal. This suggests that it is necessary but not sufficient to have a small condition number.

In order to analyze the impact of the partition choice, we need to discuss the way that the MADS algorithm handles simple bounds. MADS is applied to the reformulated Problem (4.3), in which the redundant bounds ℓ and u on z are made explicit. During the course of optimization, when MADS produces a trial point outside of these bounds, the corresponding value z_i is projected back to the boundary of the interval $[\ell_i; u_i]$. In the NOMAD implementation, this option is called

SNAP_TO_BOUNDS. With the BN converter, this happens only for the nonbasic variables which possess explicit bounds.

For HS119, 5 variables out of 16 are at one of their bounds in the optimal solution x^*. Let $\mathcal{A} := \{i \in \{1, 2, \ldots, n_x\} : x_i^* = L_i \text{ or } x_i^* = U_i\}$ denote the set of indices of active bounds. The cardinality $|I_N \cap \mathcal{A}|$ represents the number of variables in I_N that are active at the solution x^* for the choice of the partition (B, N). Recall that I_N is the set of n_z indices of the nonbasic variables of Problem (4.7). In the current problem, $|\mathcal{A}| = 5$ and $|I_N \cap \mathcal{A}|$ belongs to $\{0, 1, \ldots, 5\}$.

Figure 4.3 plots the final objective function value with respect to the condition number of B for each value of $|I_N \cap \mathcal{A}|$. The six plots illustrate the importance of the value $|I_N \cap \mathcal{A}|$. When $|I_N \cap \mathcal{A}| = 5$, all variables with an active bound are handled directly by MADS, and all runs converge to the optimal solution when the condition number is acceptable. As $|I_N \cap \mathcal{A}|$ decreases, the number of failed runs increases rapidly, even when the condition number is low.

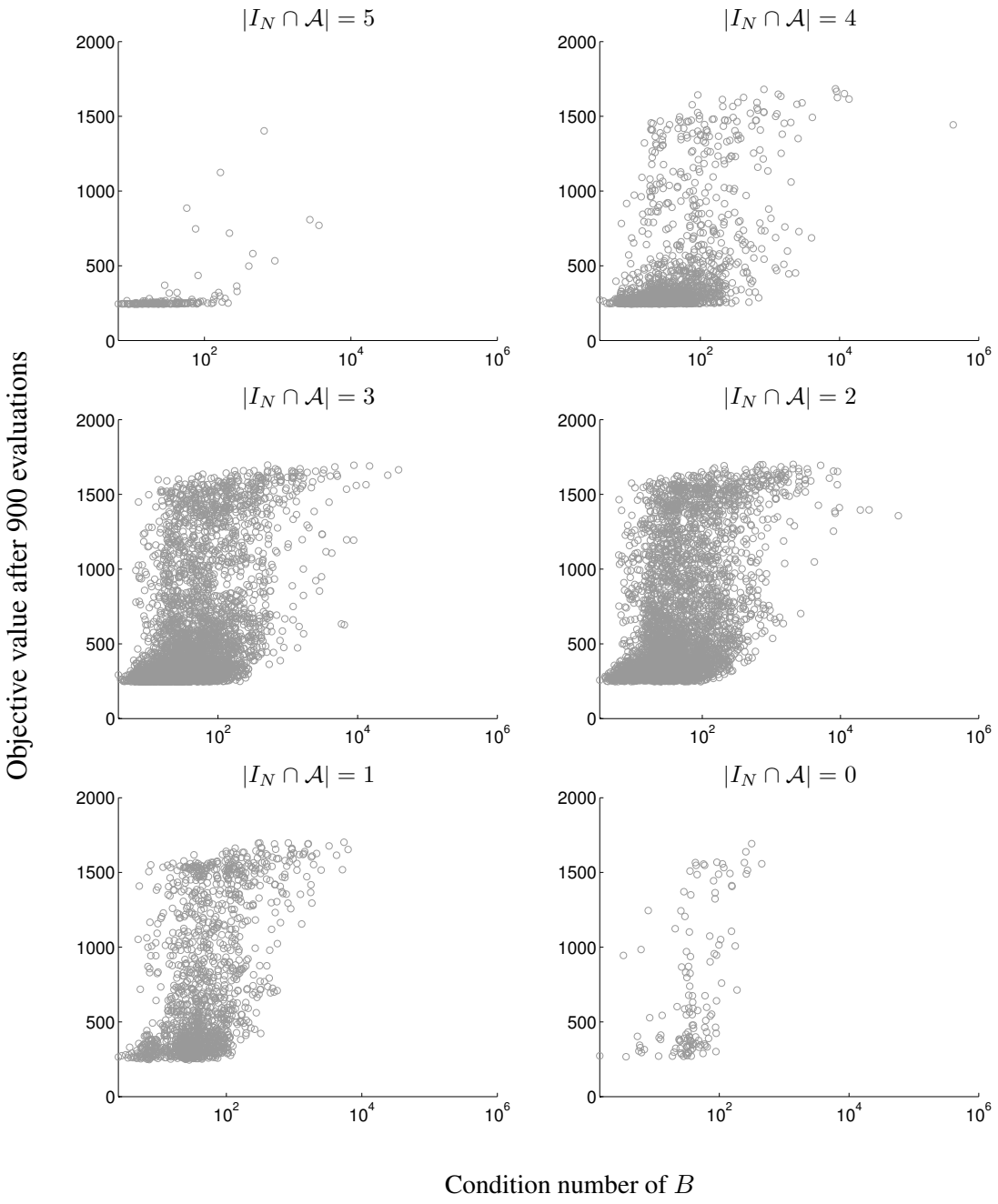


Figure 4.3 Each point represents a partition (I_B, I_N). Different graphics correspond to different values of $|I_N \cap \mathcal{A}|$
$\in \{0, 1, \ldots, 5\}$.

Indices of active bounds at the solution as well as the condition number should influence the choice of the partition (I_B, I_N). However, optimizing the condition number is an $\mathcal{NP}$-hard problem [35], and when solving an optimization problem from a starting point x^0, one does not know which bounds will be active at the optimal solution. More elaborate solutions overcoming these difficulties are proposed in Section 4.5.4 but a first method is proposed below.

The indices of the variables are sorted in increasing order with respect to the distance of the starting point to the bounds. Thus, the last indices of the ordered set will preferentially be chosen to form I_B. More precisely, the index i appears before j if the distance from x_i^0 to its closest bound is more than or equal to the distance from x_j^0 to its closest bound. When both bounds are finite, the distances are normalized by the distance between these bounds. Variables with only one finite bound come after the variables with two bounds in the sorted set. Ties are broken arbitrarily.

The columns of A are sorted in the same order as the indices of the variables and the following heuristic is applied to take into account the condition number. Construct a nonsingular matrix B' of m independent columns of A by adding the last columns of the order set. Then let c be the next column of the ordered set that does not belong to B'. Define B to be the $m \times m$ nonsingular matrix composed of columns of $B' \cup \{c\}$ that has the smallest condition number. This requires to compute m condition numbers.

4.5.3 Comparison of the four converters with HOPSPACK

A first set of numerical experiments compares the four converters BN, QR, SVD and P to the HOPSPACK software package on the 1,000 instances. In the case of the algorithm BN, the partition into the matrices B and N is done by considering the initial points and bounds, as explained in Section 4.5.2. A more extensive analysis of BN is presented in Section 4.5.4.

Comparison of the different converters with HOPSPACK is illustrated in Figure 4.4. The converter φ_P associated to the projection is dominated by all other strategies. This is not surprising since the projection does not reduce the size of the space of variables in which the optimization in performed.

When the number of function evaluations is low, it seems that HOPSPACK performs better than the other methods. However, inspection of the logs reveals that this domination is exclusive to the smallest problems, for which HOPSPACK does very well. For the larger ones, QR, SVD and BN perform better than HOPSPACK.

The figure also reveals that BN, QR and SVDclasses of converters outperform the projection, but it is not obvious to differentiate them. The next section proposes a way to combine these converters.

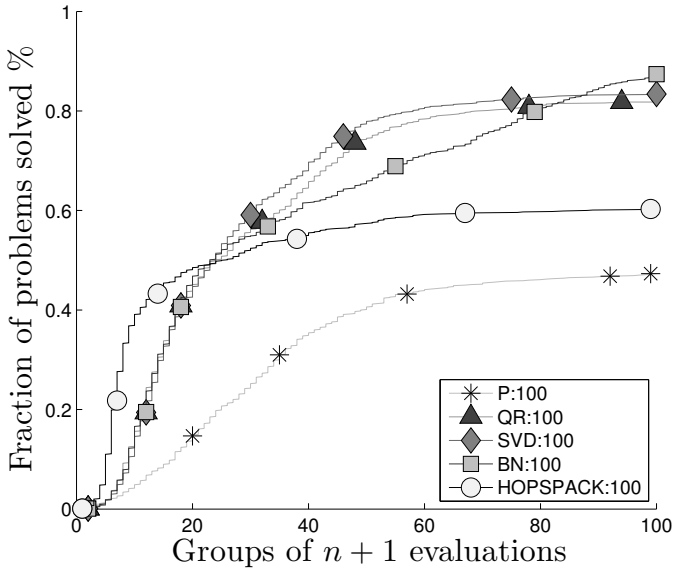


Figure 4.4 Data profiles with a relative tolerance of 0.01 for 10 problems and 100 different starting points.

4.5.4 A two-phase algorithm

The partition choice with the BN converter is crucial. Ideally, the partition should be chosen by considering the active variables at the unknown optimal solution. This section proposes a new strategy for the choice of the partition (I_B, I_N) by performing the optimization in two phases. The main idea is to initiate a descent from the starting point x^0 during a first phase, denoted Phase 1 with one of the QR, SVD or BN converters. This phase is executed with a fraction of the overall budget of function evaluations. After Phase 1, decisions are taken to choose the partition (I_B, I_N) depending on the best solution x^1 produced by Phase 1. Then a second phase, called Phase 2, launches an optimization with the corresponding BN converter using the remaining budget of function evaluations.

The point x^1 is a better candidate than x^0 to start an algorithm with the BN converter. In addition, it is possible to estimate which variables approach one of their bounds by studying the improvement performed by Phase 1 from x^0 to x^1. Thereby, unlike the choice made in Section 4.5.2 to rank the distances of each variable to its bounds, we consider the value $|x_i^0 - x_i^1|$ for each index i. Moreover, this choice provides some scaling information to NOMAD for a more efficient Phase 2.

The classes of converters used for Phase 1 and the proportions of evaluations for each phase are chosen based on numerical experiments in Section 4.5.5. The two-phase algorithm is summarized in Figure 4.5. A detailed description of the ranking method used for the choice of (B, N) and the

decision rules to determine the scale s are presented below.

Initialization
- Initial point $x^0 \in \Omega_x$
- Converter $\varphi \in \{\varphi_{QR}, \varphi_{SVD}, \varphi_{BN}\}$
- Evaluations budget η^1 and η^2 for each phase

Phase 1
- Execute MADS with converter φ
- Best value after η^1 evaluations : x^1

Choice of partition (B, N)
- if $x^0 = x^1$
  - Apply rule presented in Section 4.5.2
- else
  - Compute $\{d_i\}_{i \in \mathbb{R}^{n_x}}$ (4.8)
  - Rank the variables in a decreasing order of the d_i
  - Build B as described in the heuristic in Section 4.5.2

Scaling
- Compute the scale s as in (4.9)

Phase 2
- Execute MADS with the corresponding BN converter with scale s on the variables
- Best value after η^2 evaluations : x^2

Figure 4.5 Two-phase algorithm.

In the unlikely situation where $x^1 = x^0$, we apply the rule as defined in 4.5.2 to choose the partition. Otherwise, we apply the following rule. The only differences between this new rule and the former one is the notion of distance used to rank the variables in a decreasing order. For every $i \in \mathbb{R}^{n_x}$, a relative distance d_i is calculated for x_i^1, normalized by $|x_i^0 - x_i^1|$ when it is non-zero.

$$d_i = \begin{cases} \infty & \text{if } x_i \text{ has no finite bounds} \\ & \text{or if } x_i^0 = x_i^1 \text{ when } x_i \text{ is different from its bounds ;} \\ \frac{|x_i^1 - b_i|}{|x_i^0 - x_i^1|} & \text{if } x_i^0 \neq x_i^1 \text{ and } x_i^0 \text{ has at least a finite bound,} \\ & \text{and } b_i \text{ is its nearest bound ;} \\ 0 & \text{if } x_i^1 \text{ is equal to one of its bounds.} \end{cases} \tag{4.8}$$

The process to determine the scale s is based on the same idea used for the relative distances d_i. For every index i, $|x_i^0 - x_i^1|$ provides scaling information on the variable x_i, and the scale s_i is computed

with the following method :

$$s_i = \begin{cases} 1 & \text{if } x_i^0 = x_i^1 \text{ and } x_i \text{ has no finite bound,} \\ & \text{or if } x_i^0 = x_i^1 \text{ and } x_i \text{ is equal to one of its bounds ;} \\ \frac{1}{10}|x_i^1 - b_i| & \text{if } x_i^0 = x_i^1 \text{ and } x_i \text{ has at least one finite bound,} \\ & x_i \text{ is different from its bounds and } b_i \text{ is its nearest bound ;} \\ \frac{1}{10}|x_i^1 - x_i^0| & \text{if } x_i^0 \neq x_i^1. \end{cases} \tag{4.9}$$

In summary, Phase 2 solves the reformulated problem using the BN converter and scales the variables using the parameter s.

4.5.5 Comparison of different two-phase strategies

This section compares two-phase strategies with different converters in Phase 1 and different ponderations between the two phases.

For each class of converters QR, SVD and BN (set with the former rule defined in Section 4.5.2), we tested the two-phase strategy with the ponderation 50–50, which means that the total budget of 100(n+1) evaluations is equally shared between Phase 1 and Phase 2. Figure 4.6 reveals how the changement between each phase is beneficial, and we notice that Phase 1 works better with SVD.

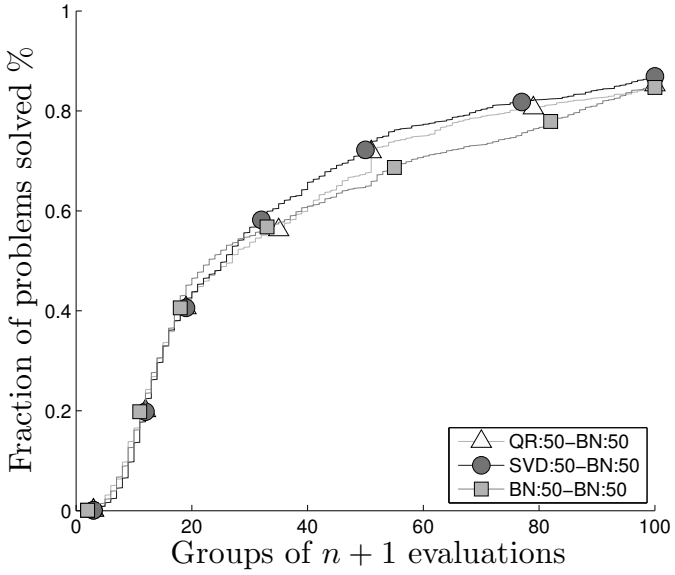


Figure 4.6 Data profiles for ponderation 50–50.

For Phase 1 using the SVD converter, different ponderations are compared in Figure 4.7. These

data profiles show that the best ponderation is 50–50. A too short Phase 1 step is inefficient because it does not lead to a good choice of BN, while a longer Phase 1 may waste evaluations.

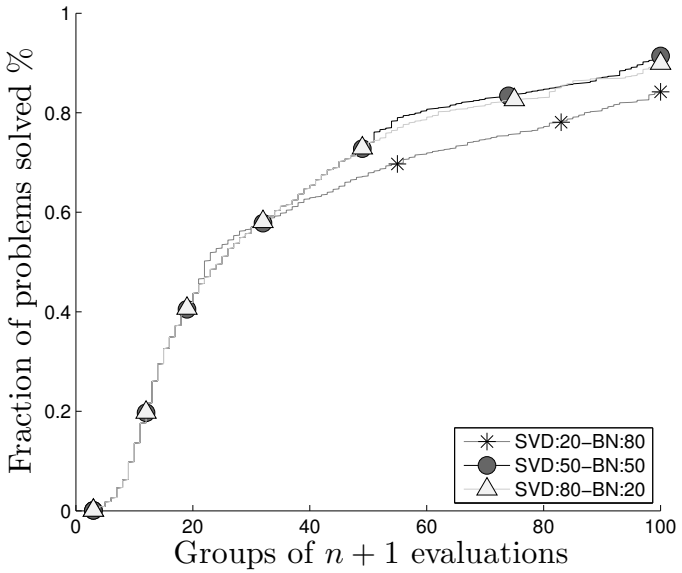


Figure 4.7 Data profiles for Phase 1 with SVD, with different ponderations.

The last comparison shown in Figure 4.8 is between the best two-phase strategy SVD :50-BN :50 and the previous best algorithms BN and SVD. We can conclude that SVD :50-BN :50 improves both algorithms.

These comparisons demonstrate that the two-phase strategy is effective, and this suggests that a new multi-phase algorithm involving more than two phases would be efficient too. We tested such a multi-phase algorithm, a four-phase method in which the two-phase algorithm SVD :25–BN :25 is repeated twice, but our results (not reported here) are not as good as expected. After analysis of these results, it appears that some changes of phase occurred too soon to be efficient and that there were issues with the control of the scaling.

We have also conducted some numerical tests with a smaller budget of evaluations. Our recommendation is to spend the first 50 groups of $n + 1$ evaluations on Phase 1 and the remaining ones on Phase 2.

4.6 Discussion

This work introduces a new generic algorithm to treat linear equalities for blackbox problems, which has been implemented with the MADS algorithm. As a result, MADS now possesses the

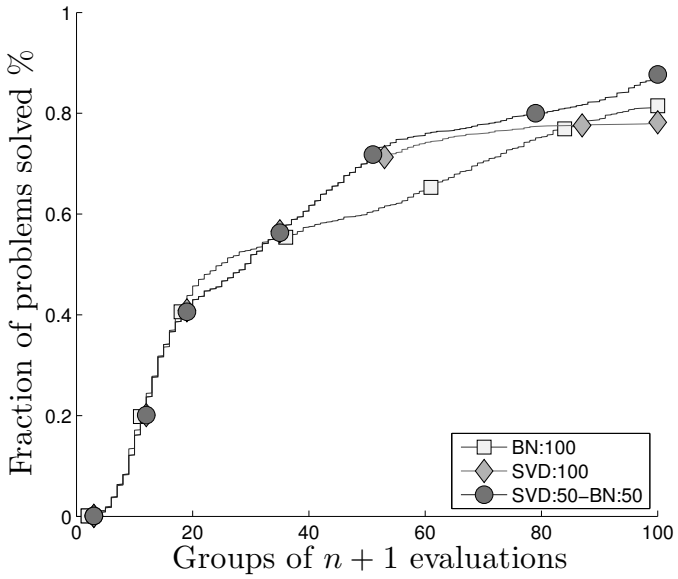


Figure 4.8 Data profiles comparing the best two-phase strategy with the previous single-phase strategies.

ability to treat linear equalities, while preserving the convergence results that the Clarke derivatives are nonnegative in the hypertangent cone relative to the nullspace of the constraint matrix. The proof relies on a theoretical result showing that hypertangent cones are mapped by surjective linear application in finite dimension.

The best strategy identified by numerical results on a collection of 10 problems from the CUTEst collection consists of first using SVD to identify potentially active variables and then continuing with BN to terminate the optimization process. This combines the advantages of both converters and is made possible by a detailed analysis of the BN converter. Our results are similar to HOPSPACK for small to medium problems, but are better for the larger instances.

Future work includes the integration of this new ability to the NOMAD software package as well as its application to linear inequalities. In addition, the inexact restoration method of [33] could rely on the present work.

4.7 Acknowledgments

The authors would like to thank two anonymous referees for their careful reading and helpful comments and suggestions.

CHAPITRE 5 ARTICLE 2 : A PROGRESSIVE BARRIER DERIVATIVE-FREE TRUST-REGION ALGORITHM FOR CONSTRAINED OPTIMIZATION

C. Audet, A. R. Conn, S. Le Digabel et Mathilde Peyrega, (2016), A progressive barrier derivative-free trust-region algorithm for constrained optimization. *Computational Optimization and Applications*, soumis le 28 juin 2016.

Abstract : We study derivative-free constrained optimization problems and propose a trust-region method that builds linear or quadratic models around the best feasible and around the best infeasible solutions found so far. These models are optimized within a trust region, and the progressive barrier methodology handles the constraints by progressively pushing the infeasible solutions toward the feasible domain. Computational experiments on smooth problems indicate that the proposed method is competitive with COBYLA, and experiments on two nonsmooth multidisciplinary optimization problems from mechanical engineering show that it can be competitive with NOMAD.

5.1 Introduction

This work targets inequality constrained optimization problems by combining ideas from unconstrained derivative-free trust-region algorithms (DFTR) [89] with the progressive barrier (PB) approach [18] to handle constraints. We consider the following optimization problem :

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & c(x) \leq 0 \\ & l \leq x \leq u \end{aligned} \tag{5.1}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a single-valued objective function, $c : \mathbb{R}^n \to (\mathbb{R} \cup \{+\infty\})^m$ corresponds to the vector of constraints, l, $u \in (\mathbb{R} \cup \{\pm\infty\})^n$ are bounds and n, $m \in \mathbb{N}$. The functions f and c are typically the outputs of a simulation, or a blackbox, and are called the true functions, while Problem (5.1) is called the true problem.

The following terminology from the taxonomy presented in [68] is used. The constraints of Problem (5.1) are assumed to be *quantifiable*, *relaxable*, *simulation*, and *known*. The taxonomy labels these assumptions as (QRSK) and have the following meaning : A relaxable constraint can be violated without causing issues in the evaluation of the objective and the other constraint functions, whereas the violation of any unrelaxable constraint makes the other outputs non exploitable. In

practice, it means that, for any algorithm, infeasible points may be considered as iterates as long as the proposed solution is feasible. Quantifiable means that the function c returns a vector of values and it is possible to measure, from an infeasible point, the distance to feasibility. Simulation means that the analytical expressions of the constraints are not available, but rather given by a simulation, and finally, there are no hidden constraints, which are constraints not known by the user.

Derivative-free methods from the literature may be classified into direct-search and model-based algorithms. At each iteration of a direct-search method, decisions for the next iterations are based only on the comparison of the newly evaluated points with the best solution so far. In model-based methods, local models are built around the current iterate. Both approaches have certain advantages. For example direct-search methods are simpler and can be more easily parallelized but on the other hand model based-methods try and account for the shape of the function more directly. Given a very badly behaved function we would use a direct-search method. If the function can be adequately approximated by a smooth function we would prefer a model-based approach unless it is essential to exploit some parallel architecture.

Many derivative-free algorithms and their applications in numerous fields are discussed in the textbook [43] and in the survey [12].

The treatment of nonlinear constraints remains a real challenge. Of course unconstrained methods can always be applied inside frameworks such as exact penalty methods [76], Lagrangian methods [28], or sequential penalty derivative-free methods [77]. Our approach is rather a direct treatment for general constraints, which only a few algorithms offer.

In model-based algorithms, the first algorithm proposed to handle general constraints without the use of their derivative is COBYLA designed by Powell and presented in [89]. It is a derivative-free trust-region algorithm and the constraints are treated in the subproblem. Linear models are built from the vertices of simplexes. In [38] a DFTR algorithm is proposed to treat problems without the use of the objective function derivatives but with the gradient of nonlinear constraints. In [104] a DFTR algorithm combines an augmented Lagrangian method with a filter technique to solve specific optimization problems presenting a separable structure. In [96] and [97], Sampaio and Toint propose to use the trust-funnel method of Gould and Toint [55] for problems with general constraints. In [102], Tröltzsch adapts SQP algorithms to general equalities in a derivative-free algorithm. In [26] general inequality constrained problems are solved by a DFTR algorithm called NOWPAC. At each iteration, the strict feasibility of the trial point is ensured, thanks to an interior path provided by a quadratic offset of the constraints. NOWPAC requires feasible initial points. Finally, in [10] and [50] DFTR algorithms using an inexact restoration method are proposed with respectively a penalty-like merit function and a filter. Algorithms treating linear constraints are proposed in [58] and [92] for model-based methods.

In the direct-search class of methods, the extreme barrier [15] treats all types of constraints by rejecting infeasible points. This is achieved by setting the objective function value to infinity at infeasible trial points. Filter methods [52] are adapted in [4], [16] and [47] to direct-search methods to treat nonlinear and quantifiable constraints. Filter methods do not combine the objective and the constraints into a single merit function as penalty-based methods, but instead they aggregate all constraints into a single constraint violation function and consider trade-offs between minimizing the objective function versus reducing the constraint violation. Kolda, Lewis and Torczon [65] present an algorithm for problems with general equality constraints where linear equalities are treated explicitly, while nonlinear constraints are handled by an augmented Lagrangian method adapted from [39] to a direct-search algorithm. Derivatives of nonlinear constraints are not used. In 2009, the progressive barrier (PB) for inequalities was proposed and specialized to the mesh adaptive direct search algorithm (MADS) [18]. The PB treats inequality constraints by aggregating constraints violations into a single function and considers trade-offs between objective function quality and feasibility. The name of the method originates from an upper bound, a threshold on the constraint violation, that is progressively reduced to force the iterates towards the feasible region. There is no need of derivatives and no penalty function. In 2010, the progressive-to-extreme barrier, [19], combines both progressive and extreme barrier techniques. Specific treatment for linear equalities or inequalities are also proposed in [24], [66], [69] and [73] for direct-search methods.

The goal of this work is to adapt the constraints handling PB technique to the DFTR framework. The document is organized as follows. Section 5.2 gives a high level description of a generic DFTR algorithm for unconstrained optimization, followed by the main PB components necessary to handle constraints. A basic framework for DFTR algorithms and a short description of the techniques used to build and improve the models are given. The PB is presented as a process to treat the constraints that allows infeasible iterates. Section 5.3 introduces a new algorithm combining the DFTR framework and the PB to solve Problem (5.1). Computational experiments are conducted in Section 5.4. Our code is shown to be competitive with COBYLA for DFO problems, and competitive with NOMAD [67] for BBO problems. We conclude with a discussion and future work in Section 5.5.

5.2 Derivative-free trust-region and progressive barrier

5.2.1 Trust-region notations and definitions

The DFTR algorithms for unconstrained optimization belong to the class of model-based algorithms, but also to the class of trust-region algorithms. The unconstrained DFTR algorithm targets

the following derivative-free optimization problem :

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{5.2}$$

As in trust-region methods, see for example [105], the algorithm works efficiently provided that one can trust the model in a neighborhood of the current point, called the trust region. At each iteration a (relatively) local model of the objective function is built and then minimized, [43]. In DFTR methods, the gradient of the objective function is unavailable to build the models, although the convergence analysis assumes some smoothness properties of the objective function and some properties of the model. Below, we define the trust region, and some notions for the models and the geometry of the sample set used to build these models. The main idea, as in classical trust-region methods with derivatives, is to solve the subproblems defined with model functions that are easier to minimize, and to perform this minimization within the trust region.

The following description is inspired from [42], with small modifications in the management of the criticality step. As in standard trust-region methods with derivatives, x^k denotes the current iterate at iteration k, and the trust region is the closed ball $B(x^k; \Delta^k) = \{x \in \mathbb{R}^n \ : \ \|x - x^k\| \leq \Delta^k\}$. The norms used in practice include Euclidian and infinite norms. The size of the trust region, within which the model is optimized, is called the trust region radius and is denoted by the scalar $\Delta^k > 0$.

In the theory developed in [43], at each iteration k the model is built through interpolation or regression from a finite set $\mathcal{Y}^k(x^k) \subset \mathbb{R}^n$ of sample points for which the objective function has been evaluated. Hence derivatives are not used to construct the model. The model of the true function f at iteration k is denoted by $\tilde{f}^k$.

The convergence analysis relies on some assumption on the quality of the models. It is convenient to assume that the functions satisfy the following definition proposed in [43] and slightly reformulated in [27]. Indeed, the definition ensures first-order convergence results similar to those of classical trust-region methods because the model has Taylor-like first-order behaviour. This definition applies in particular for continuously differentiable functions with Lipschitz continuous gradient on an appropriate open domain (see [43, chap. 6]).

Definition 5.2.1 *Let $\tilde{f}$ be a model of $f \in \mathcal{C}^1$ at $x \in \mathbb{R}^n$, and consider the radius $\Delta > 0$. Suppose that there exists a positive constant κ such that for all $y \in B(x; \Delta)$ the model $\tilde{f}$ satisfies :*

$$\begin{aligned} \|\nabla f(y) - \nabla \tilde{f}(y)\| &\leq \kappa \Delta, \\ \|f(y) - \tilde{f}(y)\| &\leq \kappa (\Delta)^2. \end{aligned}$$

Then the model $\tilde{f}$ is said to be a κ-fully-linear model of f on $B(x; \Delta)$.

These properties indicate that the model $\tilde{f}$ and its gradient are close to the function f and its gradient ∇f as long as the trust region radius is small enough. Similarities with Taylor bounds in the derivative case are clear.

The second-order convergence results require the models to satisfy the following definition :

Definition 5.2.2 *Let $\tilde{f}$ be a model of $f \in \mathcal{C}^2$ at $x \in \mathbb{R}^n$, and consider the radius $\Delta > 0$. Suppose that there exists a positive constant κ such that for all $y \in B(x;\Delta)$ the model $\tilde{f}$ satisfies :*

$$\begin{aligned} \|\nabla^2 f(y) - \nabla^2 \tilde{f}(y)\| &\leq \kappa\Delta, \\ \|\nabla f(y) - \nabla \tilde{f}(y)\| &\leq \kappa(\Delta)^2, \\ \|f(y) - \tilde{f}(y)\| &\leq \kappa(\Delta)^3. \end{aligned}$$

Then the model $\tilde{f}$ is said to be a κ-fully-quadratic model of f on $B(x;\Delta)$.

Once again similarities with Taylor-like second-order bounds can be observed. Assuming that the model satisfies stronger properties leads to stronger convergence analysis (e.g., see [26] and their definition of p-reduced fully-linear).

In [43, Chap. 6], different algorithms are detailed to construct and maintain fully-linear and fully-quadratic models. They are based on the notion of well-poisedness introduced in [43, chap. 3]. A model is said to be certifiably fully-linear (respectively certifiably fully-quadratic) when the sample set satisfies well-poisedness properties. These geometric properties on sample sets are sufficient conditions to ensure fully-linear or fully-quadratic models, and convergence is guaranteed. essentially because in the former case we know that the model improves as the trust region radius is decreased so the trust region management alone ensures that the radius stays bounded away from zero without taking any special precautions. We remark that well-poised sample sets can be determined in a finite number of steps.

5.2.2 The unconstrained DFTR algorithm

A basic framework for unconstrained DFTR algorithms is summarized in this section. The model is built using interpolation techniques rather than Taylor approximations, which implies some modification between the DFTR algorithm and a classical trust-region algorithm with derivatives.

There are different ways to define a DFTR algorithm, in particular different options to choose the sample sets and build the models [97]. To simplify and to present the main steps of the algorithm, at each iteration k, we build certifiably κ-fully-linear models for some fixed $\kappa > 0$.

At each iteration the well-poisedness of the sample set is checked and modified if necessary. In

our algorithm the sample set $\mathcal{Y}^k(x^k)$ is built by taking exactly $(n+1)$ points for linear models and $\frac{(n+1)(n+2)}{2}$ for quadratic models in a ball of radius $2\Delta^k$ around x^k.

Only interpolation techniques are used, no regression techniques are involved. If there is an insufficient number of points then additional points are randomly sampled and the geometry improvement algorithm is called, before evaluating the true function values. If there are too many points, the most recent points are chosen.

Algorithm 1 DFTR for unconstrained optimization.

Step 1 - Model construction Build a quadratic model $\tilde{f}^k$ from $\mathcal{Y}^k(x^k)$ that approximates the objective function f in $B(x^k;\Delta^k)$.
Step 2 - Subproblem Optimize the subproblem on the trust region :

$$\tilde{x}^k \in \underset{x\in B(x^k;\Delta^k)}{\operatorname{argmin}} \quad \tilde{f}^k(x). \tag{5.3}$$

The step $\tilde{s}^k = \tilde{x}^k - x^k \in B(0;\Delta^k)$ must achieve a fraction of a Cauchy step s_C^k as defined above.
Step 3 - Step calculation Evaluate f at $\tilde{x}^k$ and compute the ratio

$$\rho_f^k = \frac{f(x^k) - f(\tilde{x}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(\tilde{x}^k)}.$$

Step 4 - Trust region radius update

— If $\rho_f^k \geq \eta_1$, then set $x^{k+1} = \tilde{x}^k$ and

$$\Delta^{k+1} = \begin{cases} \gamma_{dec}\Delta^k & \text{if } \Delta^k > \mu\|g^k\|, \\ \min\{\gamma_{inc}\Delta^k, \Delta_{max}\} & \text{if } \Delta^k \leq \mu\|g^k\|. \end{cases}$$

— If $\eta_0 \leq \rho_f^k < \eta_1$, then set $x^{k+1} = \tilde{x}^k$ and

$$\Delta^{k+1} = \begin{cases} \gamma_{dec}\Delta^k & \text{if } \Delta^k > \mu\|g^k\|, \\ \Delta^k & \text{if } \Delta^k \leq \mu\|g^k\|. \end{cases}$$

— If $\rho_f^k < \eta_0$, then set $x^{k+1} = x^k$ and $\Delta^{k+1} = \gamma_{dec}\Delta^k$.

Figure 5.1 Iteration k of the DFTR unconstrained optimization algorithm.

The outline of this algorithm is presented in Figure 5.1. Additional algorithmic parameters are defined by : η_0, η_1, γ_{dec}, γ_{inc}, μ with $0 \leq \eta_0 \leq \eta_1 < 1$ (and $\eta_1 \neq 0$), $0 < \gamma_{dec} < 1 < \gamma_{inc}$, $\mu > 0$. The parameter η_1 represents a threshold to decide if the ratio comparing true and predicted improvements of the objective function is sufficiently high. The parameter η_0 is non-negative and can be

chosen equal to zero. It is introduced, among other reasons, to allow simple decrease rather than sufficient decrease. The constants γ_{dec} and γ_{inc} are multiplication factors to increase and decrease the trust region radius Δ^k. The parameter μ is a constant used to trigger the decreases of the trust region radius when the gradient becomes small, to ensure that for small values of Δ^k the difference between the true gradient (or some gradient of a nearby function) is appropriately approximated by the model gradient. A starting point x^0 must also be provided. In Step 2, the Cauchy step s_C^k is the minimizer of the model in the direction of the gradient. The notation g^k refers to the gradient of the model function $\tilde{f}^k$ and it is the direction to the Cauchy point that drives first-order descent (corresponding to the steepest descent direction in methods with derivatives).

As detailed in [42], which presents a general convergence analysis for DFTR algorithms, if the model gradient is not small and the ratio for the trust region management is large enough we increase Δ^k regardless of any other conditions, if the step is successful.

If the trust region radius is large compared to the norm of the model gradient, then the characteristics of a fully-linear or fully-quadratic model are useless as the bounds provided by the definitions may be too large to be meaningful and to guarantee an accurate model but it is important to relate this occurring to these two quantities. Under additional assumptions on f, and by replacing the original criticality step by the one described in [27], first and second order global convergence can be proved [43]. The models are fully-linear for first order analysis and fully-quadratic for second order analysis. Step 4 in Algorithm 5.1 ensures that Δ^k converges to zero and replaces both Step 5 and the criticality step in the algorithm proposed in [43, chap. 10]. Indeed, if there are infinitely many iterations in which the trust region radius is not decreased, then there are infinitely many iterations in which the value of f decreases by a minimal value, which is impossible if f is assumed to be bounded from below. A natural stopping criteria is then based on the value of Δ^k.

As the trust region radius converges to zero, it means with the definition of fully-linear and fully-quadratic models that the models become sufficiently close to the true function on the trust region. As we can prove that the model gradient converges to zero because a fraction of a Cauchy step is achieved at each iteration, the true gradient also converges to zero. In other words, convergence analysis in classical trust-region algorithm with derivative can be transferred to derivative-free trust-region algorithm under another assumption. The most important difference is that in classical trust-region algorithm with sufficient decrease, the trust region radius converges to infinity whereas it converges to zero in the DFO context. However, and contrary to [106], in the derivative case one can guarantee convergence with simple decrease if the trust region radius Δ^k goes to zero, but this may not be a good idea in practice.

An important feature of Algorithm 5.1 is that it is not required to impose fully-linear (or fully-quadratic) models at every iteration, but only at the ones where the trust region radius is decreased

or if the model gradient is (relatively) small compared with the trust region radius, and we are hoping this means that we are close to stationarity, so we have to ensure that the model gradient adequately represents the true gradient.

In this presentation the well-poisedness of the sample set is improved if needed at the beginning of each iteration. Thus the models are always certifiably fully-linear or fully-quadratic. If the well-poisedness is not guaranteed, an algorithm to improve the geometry of the sample set is called. Hence, when the ratio ρ_f^k of Step 3 is smaller than η_0, and when in addition the gradient of the model is large in comparison with the trust region radius Δ^k, we can reduce the trust region radius to improve the accuracy of the model on a smaller region. Indeed, the algorithm ensures good geometry of the sample sets at each iteration. With other management of the sample sets, a bad ratio may occur because of bad geometry and to prevent this the model improvement algorithm is needed. Reducing the trust region radius with no information regarding the properties of the models (fully-linear or fully-quadratic) is likely to slow down the algorithm. It is especially important that the gradient of the model is related to the magnitude of Δ^k when the model gradient becomes small, otherwise the Taylor-like bounds do not guarantee the accuracy of the model and the convergence criterion on the model does not imply convergence on the true function.

5.2.3 The progressive barrier

We now provide the main components used by the PB to handle constraints. Let $\mathcal{V}^k$ denote the set of all points visited by the start of iteration k, and at which the values of the true functions, f and c, have previously been evaluated. The PB method uses a constraint violation function $h : \mathbb{R}^n \to \mathbb{R}^+ \cup \{+\infty\}$ which is an aggregation of the violations of the constraint functions c_i, $i \in \{1, 2, \ldots, m\}$. It satisfies $h(x) = 0$ if and only if x is feasible. For example, one can use $h_2 = \sum_{i=1}^{m} \left(\max(c_i, 0)\right)^2$, or $h_1 = \sum_{i=1}^{m} \left(\max(c_i, 0)\right)$, or even $h_\infty = \max_{i \in \{1 \ldots m\}} (c_i, 0)$.

The PB relies on the barrier threshold h_{max}^k, which depends on k, the index of the iteration. The barrier threshold filters each point x for which $h(x) > h_{max}^k$. The idea is to obtain a feasible point by progressively decreasing the barrier threshold and selecting a point x at iteration k with a smaller constraint violation than at iteration $k-1$. However, the barrier threshold is not decreased too rapidly. Indeed, decreasing the threshold slowly allows selecting a promising infeasible point x with a low objective function value f. The sequence of thresholds $\{h_{max}^k\}_k$ is non-increasing.

The PB maintains two incumbent solutions at each iteration : a feasible x_F^k and an infeasible x_I^k one. The principle is to select these incumbents from $\mathcal{V}^k$ and to test nearby trial points in the hopes of improving the objective function value, f, or the constraint violation function value, h. We define these two points at iteration k :

— $x_F^k \in \text{argmin}\{f(x) for x \in \mathcal{V}^k \ : \ h(x) = 0\}$, the best feasible point, called the *feasible incumbent*.

— $x_I^k \in \text{argmin}\{f(x) for x \in \mathcal{V}^k \ : \ 0 < h(x) \leq h_{max}^k\}$, the infeasible point within the barrier threshold with the least objective function value, called the *infeasible incumbent*.

At least one of these two incumbents exists. When both exist, the algorithm labels one of them as the *primary incumbent* x_P^k and the other as the *secondary incumbent* x_S^k :

$$\begin{aligned} x_P^k \leftarrow x_F^k \text{ and } x_S^k \leftarrow x_I^k \qquad & \text{if } f(x_I^k) \geq f(x_F^k) - \rho|f(x_F^k)|, \\ x_P^k \leftarrow x_I^k \text{ and } x_S^k \leftarrow x_F^k \qquad & \text{otherwise,} \end{aligned}$$

where $\rho > 0$ is a trigger, set at 0.1 in our computational experiments, to favor feasible over infeasible solutions. The algorithm then explores around both incumbents, but deploys more effort around the primary one. In other words, more blackbox evaluations are computed near the incumbent that has the more promising objective function value.

If the infeasible incumbent has a lower objective function value than the feasible incumbent, i.e. $f(x_I^k) < f(x_F^k)$, then exploring near the infeasible incumbent may potentially lead to a feasible solution with a lower objective function value than $f(x_F^k)$.

Three types of iterations are distinguished, associated with different parameter update rules. These rules are based on the analysis of the set of points $\mathcal{V}^k$ and those evaluated during the iteration k. This set of points is $\mathcal{V}^{k+1}$.

— An iteration is said to be *dominating* whenever a trial point $x \in \mathcal{V}^{k+1}$ that dominates one of the incumbents is generated, i.e., when :

$$\begin{aligned} h(x) = 0 \text{ and } f(x) < f_F^k, \qquad & \text{or} \\ 0 < h(x) < h_I^k \text{ and } f(x) \leq f_I^k, \qquad & \text{or} \\ 0 < h(x) \leq h_I^k \text{ and } f(x) < f_I^k, & \end{aligned}$$

where $f_F^k = f(x_F^k)$, $f_I^k = f(x_I^k)$ and $h_I^k = h(x_I^k)$. In this case, h_{max}^{k+1} is set to h_I^k.

— An iteration is said to be *improving* if it is not dominating, but if there is an infeasible solution $x \in \mathcal{V}^k$ with a strictly smaller value of h, i.e. when :

$$0 < h(x) < h_I^k \text{ and } f(x) > f_I^k.$$

In other words, there is an infeasible solution with a lower value of h than h_I^k but a higher value of f than f_I^k. The barrier threshold is updated by setting $h_{max}^{k+1} = \max\{h(x) \ : \ h(x) <$

$h_I^k,\ x \in \mathcal{V}^k\}$. As a result, x_I^k is eliminated by the barrier threshold at iteration $k+1$. The points in $\mathcal{V}^{k+1}$ may have been generated during iteration k or during a previous iteration.

— An iteration is said to be *unsuccessful* when every trial point $x \in \mathcal{V}^k$ is such that :

$$h(x) = 0 \text{ and } f(x) \geq f_F^k, \qquad \text{or} \qquad h(x) = h_I^k \text{ and } f(x) \geq f_I^k \qquad \text{or} \qquad h(x) > h_I^k.$$

In this case, $h_{max}^{k+1} = h_I^k$ as in the dominating iteration. This implies that both incumbent solutions remain unchanged : $x_F^{k+1} = x_F^k$ and $x_I^{k+1} = x_I^k$. If the barrier threshold would be pushed further, no infeasible incumbent would be admissible as an infeasible current incumbent. Unlike the improving iteration, there are no other infeasible points to choose.

The improving iterations are the only ones which allow a reduction of the barrier threshold h_{max}^{k+1} below h_I^k. Figure 5.2 summarizes the three different cases. The leftmost figure illustrates a dominating iteration : a feasible trial point dominating x_K^k or an infeasible one dominating x_I^k is generated. The corresponding regions are represented by the thick line segment on the ordinate axis, and by the rectangular shaded region, respectively. The central figure illustrates an improving iteration : there is an $x \in \mathcal{V}^k$ whose image lies in the shaded rectangle : x has a lower constraint violation function value than h_I^k at the expense of a higher objective function value than f_I^k. Finally, the rightmost figure depicts an unsuccessful iteration. Every feasible point of $\mathcal{V}^k$ is dominated by the feasible incumbent, and every infeasible point of $\mathcal{V}^k$ has a higher constraint violation function value than h_I^k.

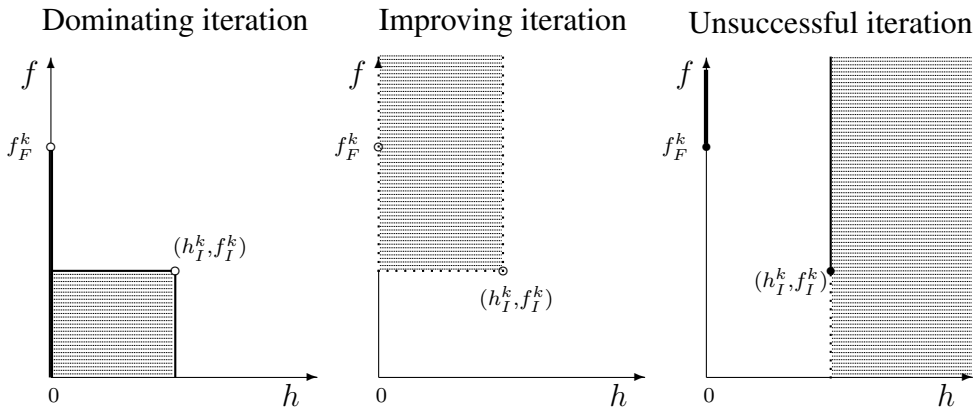


Figure 5.2 Illustration of possible regions for the 3 different types of iterations. Figure adapted from [18].

To summarize, convergence toward the feasible region is handled by the barrier threshold h_{max}^k, and selecting the infeasible point with the smallest objective function value aims at generating a solution with a good value of f.

5.2.4 The speculative line-search

In addition to the PB, we borrow a rudimentary line-search devised in the context of the MADS algorithm. The speculative search was first described in [17] as the *dynamic search*, and renamed as speculative search in [18]. The main idea of the speculative search is to explore further in a direction that leads to a dominating iteration.

More formally, in the context of the PB, let x^k be one of the incumbent solutions x_I^k or x_F^k. Suppose that exploration around x^k leads to a dominating iteration by generating the point x. Define $s = x - x^k$, the direction of success. The speculative search simply explores further along this direction at the sequence of trial points $x + 2s, x + 4s, \ldots, x + 2^j s$ and stops as soon as a trial point does not dominate the previous one.

5.3 A derivative-free trust-region algorithm using the progressive barrier

The main idea of the new algorithm to treat Problem (5.1) is to combine the unconstrained DFTR algorithm of Section 5.2.1 with constraint handling techniques from the PB of Section 5.2.3. Models of the true functions f and c are built, and two constrained subproblems based on these models are minimized at each iteration (the exact statement of these subproblems appear in Step 3 of Algorithm 2).

5.3.1 Primary and secondary subproblems

The steps in the progressive barrier derivative-free trust-region (PBTR) algorithm are similar to those of DFTR, but there are two constrained subproblems around two incumbents : the primary subproblem around the primary incumbent and the secondary subproblem around the secondary incumbent. The constraint violation threshold is managed by the PB, as in the original version of PB for MADS. The efforts in term of blackbox evaluations are different around each incumbent. Building the models around one incumbent, the primary, is made by allowing more blackbox function evaluations than around the secondary incumbent. Section 5.4 details different implementations tested, with different strategies for the allocation of evaluations between the primary and the secondary subproblems.

Thus in comparison with Algorithm 5.1, that is for unconstrained problems, Step 0 is added to determine the primary and secondary incumbent solutions and Step 7 is added to explore further along directions that lead to dominating solutions, potentially generating new incumbent solutions for the next iteration. A minimal decrease on the objective function value f is imposed to accept new points.

Some additional modifications are introduced in this hybrid algorithm. As there are two subproblems at each iteration (one around each incumbent), we define two different trust region radii at each iteration : the trust region radius Δ_F^k around x_F^k and Δ_I^k around x_I^k. The notation $\tilde{c}_i^k$ is introduced to denote the models of the contraint function c_i, $i \in \{1, 2, \ldots, m\}$. Furthermore, x_P^k, x_S^k denote the primary and secondary incumbents and $\hat{x}^k$, $\bar{x}^k$ denote the solution of the trust-region subproblems centred at the infeasible and feasible incumbents, respectively.

Recall that the set $\mathcal{V}^k \subset \mathbb{R}^n$ at iteration k corresponds to the points already evaluated by the start of iteration k. As before, the sample set of evaluated points used to build models around a point x^k is denoted by $\mathcal{Y}^k(x^k)$, and is built as presented in Section 5.2.1.

Figure 5.3 describes iteration k, with the same algorithm parameters introduced in Section 5.2.1. Recall that when the geometry of the sample set is improved it guarantees a κ-fully-linear or a κ-fully-quadratic model depending on the type of model built.

5.3.2 Progressive barrier and trust-region update rules

The update rules for h_{max}, Δ_F^k and Δ_I^k are presented below. During iteration k, the true functions f and c are evaluated at $\hat{x}^k$ generated by solving the infeasible subproblem and/or at $\bar{x}^k$ by solving the feasible one. Notice that the variables are differentiated by a hat and a bar rather than by subscripts F and I because it is possible that solving the subproblem whose domain is centered around the infeasible incumbent x_I^k might lead to a feasible solution, and solving the subproblem around x_F^k might lead to an infeasible solution. The barrier threshold h_{max}^{k+1} is updated as described in Section 5.2.3 with the following modification. The filtered set

$$\mathcal{F}^k = \{x \in \mathcal{V}^k : h(x) \leq h_{max}^k \text{ and if } x \text{ is a non-dominated point in } \mathcal{V}^k\}$$

is built and used by the PB as the replacement for $\mathcal{V}^k$. This is done because the new PBTR algorithm generates more points than MADS, and if the set of points used to update the barrier threshold h_{max}^{k+1} is not reduced, the barrier is decreased too slowly.

The trust region radius update rules of DFTR are adapted to take the constraints into account and to improve both f and h. For the objective function, we compute the same ratio ρ_f^k as in DFTR. The ratio $\hat{\rho}_f^k$ compares the relative variation of the true function f and the model $\tilde{f}^k$ at $\hat{x}^k$ and x^k, and similarly $\bar{\rho}_f^k$ compares the evaluation of f at $\bar{x}^k$ and x^k :

$$\hat{\rho}_f^k = \frac{f(x^k) - f(\hat{x})}{\tilde{f}^k(x^k) - \tilde{f}^k(\hat{x})}, \qquad \bar{\rho}_f^k = \frac{f(x^k) - f(\bar{x})}{\tilde{f}^k(x^k) - \tilde{f}^k(\bar{x})}.$$

The model $\tilde{h}^k$ of the constraint violation function h is defined using the same norm, but with the

Algorithm 2 PBTR for constrained optimization.

Step 0 - Primary and secondary incumbents Let x_P^k be the primary incumbent and x_S^k be the secondary incumbent with $\{x_P^k, x_S^k\} = \{x_F^k, x_I^k\}$

Step 1 - Construction of the sample set The sample set $\mathcal{Y}^k(x_P^k)$ is built with $\frac{(n+1)(n+2)}{2}$ points as presented in Section 5.2.1 with geometry improvement. The sample set $\mathcal{Y}^k(x_S^k)$ is built with at least 2 points.

Step 2 - Models construction Build certifiably κ-fully linear models $\tilde{f}_P^k$ and $\tilde{c}_P^k$ from $\mathcal{Y}^k(x_P^k)$ that approximate f and c in $B(x_P^k; \Delta_P^k)$.
Build linear models $\tilde{f}_S^k$ and $\tilde{c}_S^k$ from $\mathcal{Y}^k(x_S^k)$ that approximate f and c in $B(x_S^k; \Delta_S^k)$.

Step 3 - Subproblems Optimize the constrained subproblems on the trust-regions :

$$\begin{aligned} \hat{x}^k \in \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \quad & \tilde{f}_I^k(x) \\ \text{subject to} \quad & \tilde{c}_I^k(x) \leq 0 \\ & x \in B(x_I^k; \Delta(x_I^k)) \\ & l \leq x \leq u \end{aligned} \qquad \begin{aligned} \bar{x}^k \in \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \quad & \tilde{f}_F^k(x) \\ \text{subject to} \quad & \tilde{c}_F^k(x) \leq 0 \\ & x \in B(x_F^k; \Delta(x_F^k)) \\ & l \leq x \leq u. \end{aligned}$$

Step 4 - Step calculation Evaluate f and c at $\hat{x}^k$ and $\bar{x}^k$.
Compute the ratios ρ_f^k and ρ_h^k as described in Section 5.3.2.

Step 5 - Update the barrier threshold h_{max}^{k+1} The barrier threshold is updated by using PB principles and the classification of iteration in success, improvement and failure. See Section 5.3.2.

Step 6 - Update the trust region radii The trust region radii Δ_F^k and Δ_I^k are updated following rules adapted from DFTR.
See Section 5.3.2.

Step 7 - Speculative line-searches If $\hat{x}^k$ leads to a dominating iteration, perform a speculative search from x_I^k in the direction $s = \hat{x}^k - x_I^k$. If $\bar{x}^k$ leads to a dominating iteration, perform a speculative search from x_F^k in the direction $s = \bar{x}^k - x_F^k$. See Section 5.2.4.

Figure 5.3 Iteration k of the PBTR constrained optimization algorithm.

model of c. For example, if $h = \sum_{i=1}^{m} (\max(c_i, 0))^2$, then $\tilde{h}_2^k = \sum_{i=1}^{m} \left(\max(\tilde{c}_i^k, 0)\right)^2$. The second set of ratios compare the variation of the true function h over the variation of the model function $\tilde{h}^k$:

$$\hat{\rho}_h^k = \frac{h(x^k) - h(\hat{x}^k)}{\tilde{h}^k(x^k) - \tilde{h}^k(\hat{x}^k)}, \qquad \bar{\rho}_h^k = \frac{h(x^k) - h(\bar{x}^k)}{\tilde{h}^k(x^k) - \tilde{h}^k(\hat{x}^k)}.$$

The new rules are an adaptation of the DFTR update rules, with three main differences. First, the ratio for the constraint violation function h is taken into account. Second, there are two incumbents : a feasible and an infeasible one. Third, the condition to increase the trust region radius in Step 4 of Algorithm 5.1 is that the trust region radius is small compared to the norm of the model gradient of the objective function. In the constrained case, we introduce two Boolean variables defined as follows :

$$\mathcal{P}_I^k \text{ is true iff } \|\nabla \tilde{h}_I^k(x_I^k)\| \geq \mu \Delta_I^k \qquad \text{and} \qquad \mathcal{P}_F^k \text{ is true iff } \|\nabla \tilde{f}_F^k(x_F^k)\| \geq \mu \Delta_F^k$$

These Boolean variables are used to devise the trust region radius update rules. In what follows, the negation of $\mathcal{P}_I^k$ is denoted by $\neg\mathcal{P}_I^k$.

For each trust region radius Δ_I^k and Δ_F^k the update rule depends on the status of feasibility of the new generated points $\hat{x}^k$ and $\bar{x}^k$ respectively. The update rules are similar to that of DFTR, with some modifications : the ratios for both f and h are taken into account when the new generated point is infeasible.

— Update rule for Δ_I^k when $\hat{x}^k$ is infeasible :

$$\Delta_I^{k+1} = \begin{cases} \gamma_{inc}\Delta_I^k & \text{if } \eta_1 \leq \hat{\rho}_f^k \text{ and } \eta_1 \leq \hat{\rho}_h^k \text{ and } \mathcal{P}_I^k, \\ \gamma_{dec}\Delta_I^k & \text{if } \hat{\rho}_f^k < \eta_0 \text{ or } \hat{\rho}_h^k < \eta_0 \text{ or } \neg\mathcal{P}_I^k, \\ \Delta_I^k & \text{otherwise.} \end{cases} \tag{5.4}$$

— Update rule for Δ_I^k when $\hat{x}^k$ is feasible :

$$\Delta_I^{k+1} = \begin{cases} \gamma_{inc}\Delta_I^k & \text{if } \eta_1 \leq \hat{\rho}_f^k \text{ and } \mathcal{P}_I^k, \\ \Delta_I^k & \text{if } \eta_0 \leq \hat{\rho}_f^k < \eta_1 \text{ and } \mathcal{P}_I^k, \\ \gamma_{dec}\Delta_I^k & \text{otherwise.} \end{cases} \tag{5.5}$$

— Update rule for Δ_F^k when $\bar{x}^k$ is infeasible :

$$\Delta_F^{k+1} = \begin{cases} \gamma_{inc}\Delta_F^k & \text{if } \eta_1 \leq \bar{\rho}_f^k \text{ and } \eta_1 \leq \bar{\rho}_h^k \text{ and } \mathcal{P}_F^k, \\ \gamma_{dec}\Delta_F^k & \text{if } \bar{\rho}_f^k < \eta_0 \text{ or } \bar{\rho}_h^k < \eta_0 \text{ or } \neg\mathcal{P}_F^k, \\ \Delta_F^k & \text{otherwise.} \end{cases} \tag{5.6}$$

— Update rule for Δ_F^k when $\bar{x}^k$ is feasible :

$$\Delta_F^{k+1} = \begin{cases} \gamma_{inc}\Delta_F^k & \text{if } \eta_1 \leq \bar{\rho}_f^k \text{ and } \mathcal{P}_F^k, \\ \Delta_F^k & \text{if } \eta_0 \leq \bar{\rho}_f^k < \eta_1 \text{ and } \mathcal{P}_F^k, \\ \gamma_{dec}\Delta_F^k & \text{otherwise.} \end{cases} \tag{5.7}$$

In every case, when both ratios exceed η_0, but at least one of them is less than η_1, then the trust region radius remains unchanged at iteration $k+1$. Indeed, it is analogous to the unconstrained algorithm, in which a ratio between η_0 and η_1 implies no modification for the trust region radius.

When the new generated point is feasible, the update rules are defined by considering only the ratio of f, and similar rules as in the unconstrained case are applied.

5.3.3 Convergence analysis

The goal of this section is to demonstrate that at least one sequence among the two trust region radii sequences converges to zero.

We suppose in the following that the functions f and c_i for $i \in \{1, .., m\}$ are twice continuously differentiable and that the function f is bounded from below. It is also assumed that at each iteration, the feasible and infeasible subproblems are solved by doing at least a fraction of the Cauchy step for f in the case of the feasible subproblem, and for h in the case of the infeasible subproblem. Let denote by κ_{Cauchy} the fraction of Cauchy decrease (one for all the functions) and by κ_H the bound on the Hessian of the models supposed uniformly bounded. As in [42] the following assumptions are made :

Assumptions 1 *Consider f, $c \in \mathcal{C}^2$, where f is bounded from below. For every iteration k, it is possible to compute $\hat{x}$ (if x_I^k exists) and $\bar{x}$ (if x_F^k exists) such that :*

$$\tilde{h}(x_I^k) - \tilde{h}^k(\bar{x}) \geq \frac{\kappa_{Cauchy}}{2}\|g_h^k\| \min\left\{\frac{\|g_h^k\|}{\kappa_H}, \Delta_I^k\right\}$$

and

$$\tilde{f}(x_F^k) - \tilde{f}^k(\bar{x}) \geq \frac{\kappa_{Cauchy}}{2}\|g_f^k\| \min\left\{\frac{\|g_f^k\|}{\kappa_H}, \Delta_F^k\right\},$$

where g_f^k and g_h^k are respectively the model gradient of f at x_F^k and the model gradient of h at x_I^k.

The assumption combines those involving the Cauchy steps and the uniform bound on the model Hessians. The assumption that f is bounded from below, and the fact that the speculative line-search requires minimal decrease on f implies that the line-search will necessarily terminate after

a finite number of steps.

The following pair of theorems shows that at least one trust region radii sequences converges to 0.

Theorem 6 *Let Δ_F^k be the sequence of trust region radii around the feasible incumbents produced by Algorithm* PBTR *under Assumption 1. If the algorithm generates at least one feasible solution, then*

$$\lim_{k \to +\infty} \Delta_F^k = 0$$

PROOF. Suppose that the algorithm PBTR generates a first feasible solution at iteration k_0. It follows that all subsequent iterations $k > k_0$, will have a feasible incumbent solution, and the sequence $\{\Delta_F^k\}$ is well-defined.

Suppose that the sequence $\{\Delta_F^k\}$ does not converge to zero. Then there exists an $\varepsilon > 0$ such that the cardinality of the set $\{k : \Delta_F^k > \varepsilon\}$ is infinite. Now, since $\gamma_{inc} > 1$, the cardinality of the set $\{k : \Delta_F^{k+1} > \Delta_F^k > \frac{\varepsilon}{\gamma_{inc}}\}$ is also infinite, implying that there is an infinite number of iterations k where Δ_F^k is not decreased, i.e. where $\bar{\rho}_f^k \geq \eta_0$.

Assumption 1 implies that a fraction of the Cauchy step for f is achieved :

$$f(x_F^k) - f(\bar{x})) \geq \eta_0(\tilde{f}^k(x_F^k) - \tilde{f}^k(\bar{x})) \geq \eta_0 \frac{\kappa_{Cauchy}}{2} \|g_f^k\| \min\left\{ \frac{\|g_f^k\|}{\kappa_H}, \Delta_F^k \right\} \geq C > 0$$

where $C = \eta_0 \frac{\kappa_{Cauchy}}{2} \min\left\{\frac{\mu}{\kappa_H}, 1\right\} \mu \frac{\varepsilon^2}{\gamma_{inc}^2}$, because for those iterations k,

$$\|g_f^k\| \geq \mu \Delta^k > \mu \frac{\varepsilon}{\gamma_{inc}}.$$

As f is bounded from below and the objective function value is decreased an infinite number of times by at least the constant $C > 0$, there is a contradiction. Therefore $\Delta_F^k \to 0$. □

Theorem 7 *Let Δ_I^k be the sequence of trust region radii around the infeasible incumbents produced by Algorithm* PBTR *under Assumption 1. If the algorithm never generates any feasible solutions, then*

$$\lim_{k \to +\infty} \Delta_I^k = 0.$$

PROOF. Suppose that the algorithm PBTR never generates any feasible solutions. This implies that the initial point x^0 is infeasible. Consider the two cases. (i) If iteration k is either dominating or unsuccessful, then $h_{max}^{k+1} = h(x_I^k)$ and therefore x_I^k is an infeasible incumbent candidate at iteration $k + 1$. (ii) If the iteration is improving, then by definition there exists an infeasible point with a

better value of h than $h(x_I^k)$, and the update of h_{max}^{k+1} allows to choose this point. By induction all incumbents are infeasible, and the sequence Δ_I^k is well-defined for all $k \geq 0$.

Similar arguments from the proof of the previous theorem can be applied here. Suppose that the sequence $\{\Delta_I^k\}$ does not converge to zero. Then there exists an $\varepsilon > 0$ such that the cardinality of the set $\{k : \Delta_I^{k+1} > \Delta_I^k > \frac{\varepsilon}{\gamma_{inc}}\}$ is infinite. There is then an infinite number of iterations k where Δ_I^k is not decreased, which means that $\hat{\rho}_h^k \geq \eta_0$. Assumption 1 implies that a fraction of the Cauchy step for f is achieved :

$$h(x_I^k) - h(\hat{x})) \geq \eta_0(\tilde{h}^k(x_I^k) - \tilde{h}^k(\hat{x})) \geq \eta_0 \frac{\kappa_{Cauchy}}{2} \|g_h^k\| \min\left\{\frac{\|g_h^k\|}{\kappa_H}, \Delta_I^k\right\} \geq C > 0$$

where $C = \eta_0 \frac{\kappa_{Cauchy}}{2} \min\left\{\frac{\mu}{\kappa_H}, 1\right\} \mu \frac{\varepsilon^2}{\gamma_{inc}^2}$, because for those iterations k,

$$\|g_h^k\| \geq \mu \Delta^k > \mu \frac{\varepsilon}{\gamma_{inc}}.$$

As h is bounded from below by the value 0, and the constraint violation function value is decreased an infinite number of times by at least the constant $C > 0$, there is a contradiction. Therefore $\Delta_I^k \to 0$. □

There are two possible outcomes for Algorithm PBTR. Either it generates a feasible solution, in which case $\Delta_F^k \to 0$, or either it does not, in which case $\Delta_I^k \to 0$. At least one trust region radii sequences converges to 0. However, if the algorithm generates both feasible and infeasible points, then nothing can be said about the limit of the sequence Δ_I^k.

There are examples when the sequence Δ_I^k can diverge. There are examples when the sequence Δ_I^k can converge, in some cases to a nonzero value, and in other cases Δ_I^k can converge to 0.

5.4 Implementation and computational results

This section describes our Python implementation of the PBTR algorithm, and shows computational experiments comparing it with two state-of-the-art software packages. We first describe our computational testbed.

5.4.1 Computational testbed

The computational results are generated using 40 small-scale DFO analytical problems from the CUTEst collection [54] which respect the form of Problem (5.1), with inequality constraints and

bounds. The unscaled problems from CUTEst are not selected because our implementation does not incorporate dynamic scaling as done in COBYLA from NLOPT and NOMAD version 3.7.2.

For each analytical problem the initial point provided by CUTEst is used. It lies inside the bounds but does not necessarily satisfy the other constraints. Each instance is considered with a budget of $100(n+1)$ blackbox evaluations, the same order of magnitude used in [22] and [24]

The name, number of variables (n), number of constraints (m) and information about these problems are given in Table 5.1.

Two derivative-free multidisciplinary design optimization (MDO) problems from mechanical engineering are also used in a second round of computational experiments.

The first problem is called `AIRCRAFT_RANGE` and is taken from [99]. Computational experiments on this problem are conducted in [13, 19, 85]. Three coupled disciplines, namely structure, aerodynamics and propulsion are used to represent a simplified aircraft model with 10 variables. The objective function is to maximize the aircraft range under bounds constraints and 10 relaxable constraints. The blackbox implements a fixed-point method through the different disciplines in order to compute the different quantities.

The second problem is called `SIMPLIFIED_WING` and aims at minimizing the drag of a wing by optimizing its geometry [101] through 7 bound-constrained variables, subject to 3 relaxable constraints. This multidisciplinary design optimization problem involves structures and aerodynamics. Both `AIRCRAFT_RANGE` and `SIMPLIFIED_WING` are initialized with a point chosen in the center of the region defined by the bounds.

Data profiles [83] are used to illustrate the results on the analytical problems. These graphs allow to compare different algorithms on a set of instances given a tolerance parameter $\tau \in [0; 1]$, fixed to 10^{-3} in this section. More precisely, a curve is associated to each method in a $x - y$ plane, where y corresponds to the proportion of problems close within τ to a reference solution, after x groups of $n + 1$ evaluations have been done. This reference solution is the best solution achieved by the different methods that are plotted in the graph. Data profiles were originally introduced for unconstrained problems, and have been adapted here to the constrained case, by considering only feasible solutions. With this strategy, it may occur though that no algorithm solves a problem when no feasible solutions have been found. A tolerance of 10^{-14} for h is used to consider a point as being feasible.

Performance profiles from [83] are also plotted. For such graphs, a performance ratio $r_{p,s}$ is defined by

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} \ : \ s \in S\}}$$

Table 5.1 Description of the 40 analytical problems.

Name	n	m	lower bounds	upper bounds	initial point
avgasb	8	10	8	8	Feasible
b2	3	3	0	0	Infeasible
chaconn1	3	3	0	0	Infeasible
himmelp5	2	3	2	2	Infeasible
hs10	2	1	0	0	Infeasible
hs11	2	1	0	0	Infeasible
hs12	2	1	0	0	Feasible
hs15	2	2	0	1	Infeasible
hs18	2	2	2	2	Infeasible
hs19	2	2	2	2	Infeasible
hs22	2	2	0	0	Infeasible
hs23	2	5	2	2	Infeasible
hs24	2	3	2	0	Feasible
hs29	3	1	0	0	Feasible
hs30	3	1	3	3	Feasible
hs31	3	1	3	3	Feasible
hs33	3	2	3	1	Feasible
hs34	3	2	3	3	Feasible
hs35	3	1	3	0	Feasible
hs36	3	1	3	3	Feasible
hs43	4	3	0	0	Feasible
hs57	2	1	2	0	Feasible
hs64	3	1	3	0	Infeasible
hs72	4	2	4	4	Infeasible
hs76	4	3	4	0	Feasible
hs84	5	6	5	5	Feasible
hs86	5	10	5	0	Feasible
hs95	6	4	6	6	Infeasible
hs96	6	4	6	6	Infeasible
hs97	6	4	6	6	Infeasible
hs98	6	4	6	6	Infeasible
hs100	7	4	0	0	Feasible
hs101	7	6	7	7	Infeasible
hs108	9	13	1	0	Infeasible
kiwcresc	3	2	0	0	Infeasible
lootsma	3	2	0	1	Feasible
polak6	5	4	0	0	Infeasible
simpllpb	2	3	0	0	Infeasible
snake	2	2	0	0	Infeasible
spiral	3	2	0	0	Feasible

for Algorithm s on Problem p where S is the set of algorithms tested. It is the ratio of the number of evaluations needed to solve the problem ($t_{p,s}$) over the number of evaluations needed to solve it with the best algorithm. The performance profile $P_s(\alpha)$ of Solver $s \in S$ corresponds to the probability for s to have a performance ratio within a factor $\alpha \in \mathbb{R}$. It is approximated by

$$P_s(\alpha) = \frac{1}{n_p} size\{p \in \mathcal{P} \ : \ r_{p,s} \leq \alpha\},$$

where n_p is the number of problems and $\mathcal{P}$ the set of problems. The curve on the far left side of a performance profile begins with $\alpha = 1$ and indicates the ratio of problems solved by s. The curve when $\alpha \to \infty$ indicates on how many problems s converges. Data profiles and performance profiles are complementary.

Convergence graphs (objective value versus number of evaluations) are plotted for the two MDO problems. The NOMAD (version 3.7.2) and COBYLA (NLOPT) software packages are used with their default settings. NOMAD with default settings uses quadratic models as presented in [40]. The software COBYLA implements a derivative-free trust-region algorithm as described in [89]. The models used are linear and a penalty function is used to handle the constraint, based on the infinity norm.

5.4.2 An implementation of the PBTR algorithm

We now describe the implementation specifics of PBTR. Different options for the management of the sample sets are proposed. Finally a variant of the update rule for the barrier threshold, h_{max}, is presented, giving two options for this update, the original as in MADS and a new one.

Sample set management and subproblems

The techniques used to build the sample sets for linear and quadratic models are detailed in Section 5.2.1. The PBTR algorithm distinguishes two incumbents, the primary and the secondary. Three different options are implemented to manage the sample sets of these incumbents. Each builds the sample set of points around a point x by taking every point at a distance within twice the size of the trust region radius. If that set of points is too large, the more recently generated ones are selected. Furthermore, at each iteration the geometry improvement algorithm is called for both the sample sets built around the primary incumbent and the secondary incumbent. The subproblems are optimized with a limit of 100 iterations of Ipopt.

Ipopt is a nonlinear solver using an interior point method. Ipopt is used with a tolerance for each constraint equal to 10^{-8}, which is compliant with a global tolerance for the problem defined by

$h_2(x) < 10^{-14}$.

The first option is named QUAD_QUAD. Quadratic models are built around both the primary and secondary incumbents x_P^k and x_S^k. The sample set around the primary iterate contains exactly $\frac{(n+1)(n+2)}{2}$ points and then the models built are completely determined. To have this exact number of points, new points are sampled and evaluated by respecting the well-poisedness of the sample set. The models are built with interpolation and are completely determined by the points in the sample sets. The sample set around the secondary iterate contains at most $\frac{(n+1)(n+2)}{2}$ points depending if there are less than $\frac{(n+1)(n+2)}{2}$ points in a ball of radius $2\Delta_S^k$ around the secondary iterate. Then the models built are underdetermined or completely determined depending on the number of points. In the underdetermined case the minimum of the Frobenius norm is taken as explained in [43, chap. 5]. More precisely, among all quadratic interpolation functions that pass trough the sample set, we select the one with the least Frobenius norm of the Hessian matrix of the quadratic function.

The second option is named QUAD_LIN. Quadratic models are built around the primary incumbents and linear models around the secondary incumbents. As above, the primary sample set contains exactly $\frac{(n+1)(n+2)}{2}$ points and the models built are completely determined. The secondary sample set contains at most $n+1$ points depending if there are less than $(n+1)$ points in a ball of radius $2\Delta_S^k$ around the secondary iterate.

The third option is named LIN_LIN. Linear models are built around both primary and secondary incumbents x_P^k and x_S^k. The sample set around the primary iterate contains exactly $n+1$ points and then the models built are completely determined. The sample set around the secondary iterate contains at most $n+1$ points.

These three options for the management of the sample set are compared in Section 5.4.3.

Revised barrier threshold update rules

Recall that an improving iteration happens when at the end of an iteration, there are no points dominating the current incumbents, but there is at least one infeasible point which is not dominated by the infeasible incumbent. It means that there is at least one point x such that $f(x) > f(x_I^k)$ and $0 < h(x) < h(x_I^k)$.

A revised version of the barrier threshold update rule is proposed in case of an improving iteration. In the original version of the progressive barrier in MADS, as presented in Section 5.2.3, after an improving iteration k, the barrier threshold is set to the value $h_{max}^{original} = \max\{h(x) : h(x) < h_I^k,\ x \in \mathcal{V}^k\}$. Selecting $h_{max}^{original}$ to update the barrier threshold is appropriate in the context of MADS but not in the context of a DFTR algorithm. The reason is that to construct models, DFTR spends more function evaluations at every iterations than MADS, and the barrier threshold parame-

ter would converge very slowly to zero.

To circumvent this undesirable behaviour, we define x_*^k as the infeasible point with the best value of h at the end of iteration $k : x_*^k \in \operatorname{argmin}\{h(x) : h(x) > 0,\ x \in \mathcal{F}^k\}$. and propose the following rule to reduce the threshold parameter : Following an improving iteration, we set

$$h_{max}^{k+1} = 0.9 \times h_{max}^{original} + 0.1 \times h(x_*^k).$$

This update rule guaranties to find at iteration $k + 1$ an infeasible incumbent satisfying the barrier threshold. And it offers a trade-off to push the barrier threshold adequately, between a too aggressive strategy decreasing the barrier threshold too rapidly and an unaggressive strategy susceptible to fail to find a feasible point.

In the computational experiments below, the two barrier threshold update rules are labelled by ORIGINAL and REVISED.

5.4.3 Computational comparisons of different strategies for PBTR

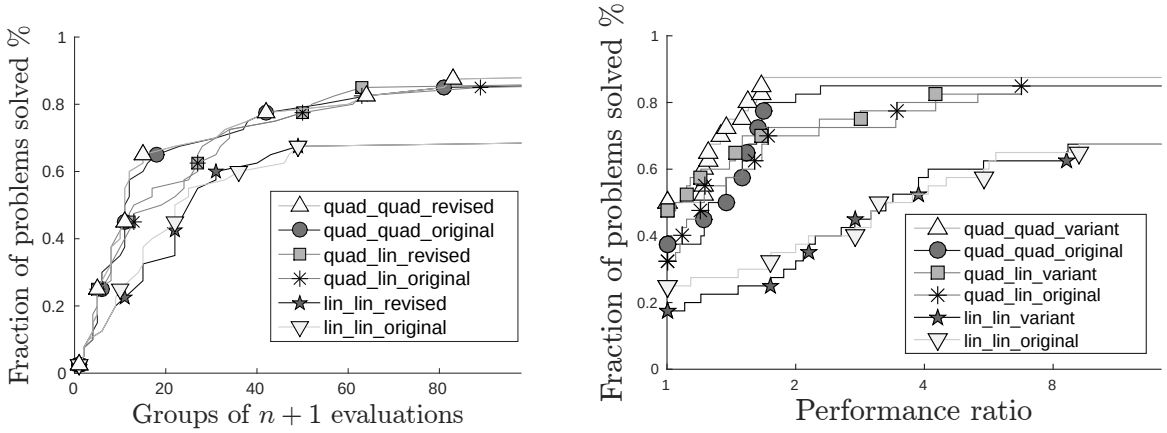
The testbed used is described and different options implemented to test versions of PBTR are described above. In addition, other solvers exist to solve Problem (5.1). Here computational results are presented to determine the best strategy for PBTR and its validity in existing algorithms.

Figure 5.4 compares the six possible strategies by plotting the proportion of problems solved versus the number of groups of $n + 1$ evaluations. Figure 5.5 compares specifically the two quadratic strategies, QUAD_QUAD ORIGINAL and REVISED, by plotting the proportion of problems solved versus the number of groups of $n + 1$ evaluations. The conclusions that are drawn do not take into account the time necessary to construct the linear and quadratic models. The figures reveal that combining quadratic models in both primary and secondary subproblems is the most efficient, and that the revised update rule for the barrier threshold performs better. Hence the best strategy is QUAD_QUAD_REVISED. The second best strategy also builds both quadratic models, but uses the original threshold update rule. At least for the problems tested, the improvements made at each iteration with quadratic models are worth the cost of building these models.

5.4.4 Comparison of PBTR with NOMAD and COBYLA

In order to validate our algorithm the best strategy for PBTR (QUAD_QUAD with REVISED), is compared to state-of-the art software packages NOMAD and COBYLA.

The data profiles in Figure 5.6(a) shows that our algorithm is competitive with COBYLA on the benchmark set of smooth analytical problems. As expected, both model-based COBYLA and PB-



(a) Data profiles with $\tau = 10^{-3}$. (b) Perf. profiles at $100(n+1)$ evaluations.

Figure 5.4 Data profiles for six PBTR strategies.

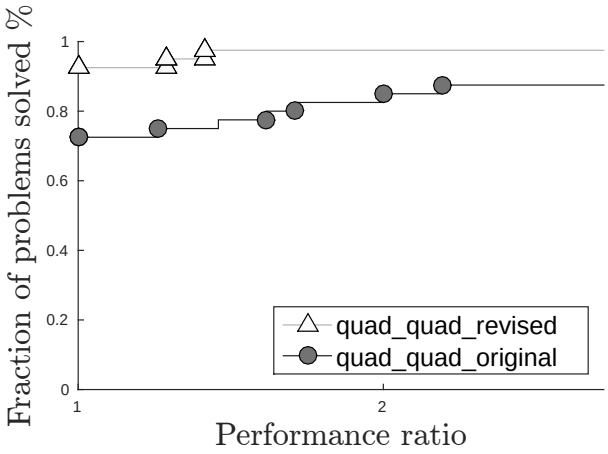
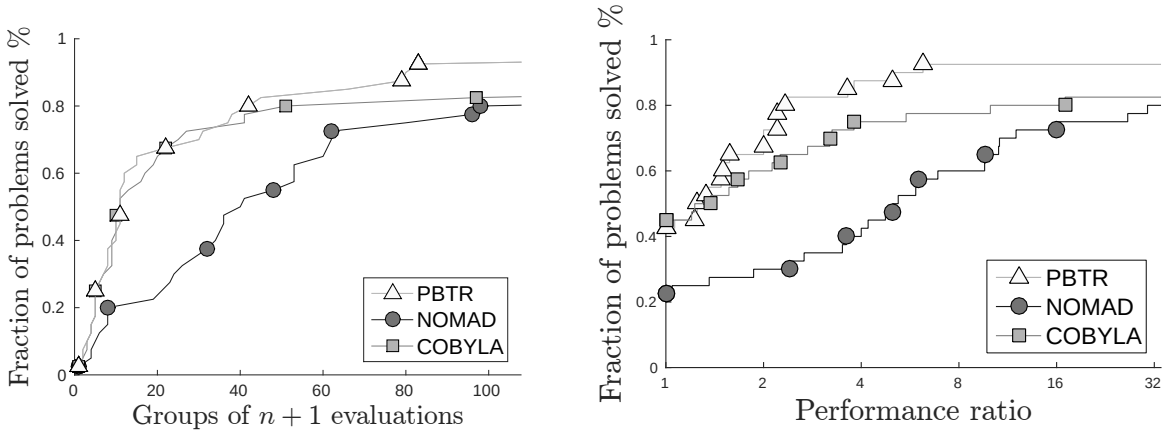


Figure 5.5 Data profiles comparing the original and revised quadratic PBTR strategies.

TRQUAD_QUAD perform better than the direct-search NOMAD algorithm. This behaviour was anticipated, as we do not recommend to use a direct-search method for problems that can be well-approximated by smooth functions. The performance of PBTR is comparable to that of COBYLA, and when the number of function evaluations exceeds $40(n+1)$, PBTR slightly outperforms COBYLA. This last observation is confirmed by the performance profiles in Figure 5.6(b.)



(a) Data profiles with $\tau = 10^{-3}$.

(b) Perf. profiles at $100(n+1)$ evaluations.

Figure 5.6 Comparison of PBTR with NOMAD and COBYLA on DFO problems.

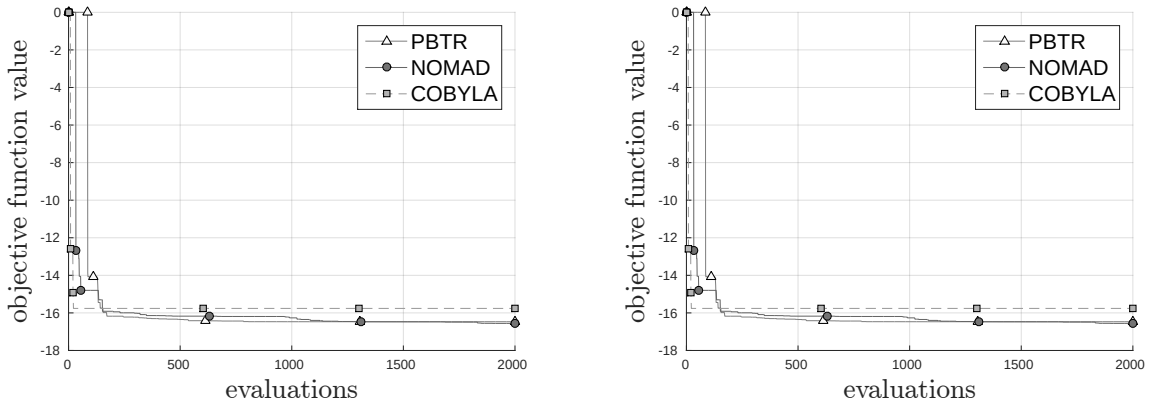
The convergence graphs for both blackbox multidisciplinary design optimization problems are plotted in Figure 5.7. For the AIRCRAFT_RANGE problem, the solver COBYLA decreases quickly but stalls at a feasible solution with objective function value around -1600, whereas both NOMAD and PBTR converge to solutions with a similar objective function value near -4000. This can be explained by the fact that COBYLA is, with its linear models, a first order algorithm. The plot reveals that NOMAD improves the solution more rapidly than PBTR on this BBO problem. The right part of the figure on the SIMPLIFIED_WING problem indicates a similar behaviour for COBYLA.

However, here both NOMAD and PBTR behave in a very similar way : both convergence graph overlap and reach the same objective function value.

5.5 Discussion

This work shows how to treat nonlinear inequalities for derivative-free and blackbox optimization problems, by combining techniques from derivative-free trust-region methods with the progressive barrier strategy. After MADS, it is the first algorithm to deploy the progressive barrier.

Different strategies are compared and the best one is identified by computational results on a collection of 40 problems from the CUTEst collection. It consists of building quadratic models for



(a) AIRCRAFT_RANGE. (b) SIMPLIFIED_WING.

Figure 5.7 Convergence graphs of PBTR, NOMAD and COBYLA for the two MDO problems.

every subproblem solved and of a trade-off rule to update the barrier threshold. This new algorithm PBTR combines features of both model-based and direct-search algorithms. Our computational results suggest that PBTR is competitive with COBYLA and preferable to NOMAD on analytical DFO problems, and that PBTR is competitive with NOMAD and preferable to COBYLA on nonsmooth blackbox optimization problems.

Future work includes the integration of dynamic scaling in our implementation and other sample set managements, to allow overdetermined models, or to numerically analyze the frequency to improve the geometry of sample sets. Penalty function could also be examined for the subproblem treatment. Finally, prior to our work, the PB was only adapted to the MADS algorithm. We have shown that it can also be successfully adapted to a trust-region algorithm. Adaptations to other nonlinear algorithms should also be investigated.

CHAPITRE 6 ARTICLE 3 : A DERIVATIVE-FREE TRUST-REGION AUGMENTED LAGRANGIAN ALGORITHM

C. Audet, S. Le Digabel et Mathilde Peyrega, (2016), A derivative-free trust-region augmented Lagrangian algorithm. *Optimization Letters*, soumis le 5 juillet 2016.

Abstract : We present a new derivative-free trust-region (DFTR) algorithm to solve general nonlinear constrained problems with the use of an augmented Lagrangian method. No derivatives are used, neither for the objective function nor for the constraints. An augmented Lagrangian method, known as an effective tool to solve equality and inequality constrained optimization problems with derivatives, is exploited to minimize the subproblems, composed of quadratic models that approximate the original objective function and constraints, within a trust region. The trust region ratio which follows the classical update rules for the trust region radius is defined by comparing the true decrease of the augmented Lagrangian merit function with the expected decrease. This mechanism allows to reuse the basic unconstrained DFTR update rules with minor modifications. Computational experiments on a set of analytical problems suggest that our approach outperforms HOPSPACK and is competitive with COBYLA. Using an augmented Lagrangian, and more generally a merit function, to design the DFTR update rules with constraints is shown to be an efficient technique.

6.1 Introduction

We consider the general optimization problem

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{subject to} & h(x) = 0 \\ & g(x) \leq 0 \\ & l \leq x \leq u \end{array} \tag{6.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a single-valued function, $h : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^p$ correspond to equality and inequality constraints, l, $u \in (\mathbb{R} \cup \{\pm\infty\})^n$ are bounds and n, m, $p \in \mathbb{N}$ are the dimension of the problem, the number of equalities and inequalities, respectively. We are interested in a class of derivative-free optimization (DFO) problems in which the functions f, h and g are twice continuously differentiable, but their derivatives are unavailable. Such a situation may occur, for example, when a simulation is involved, for which it is difficult or costly to estimate derivatives.

Algorithms for problems of the form (6.1) exist. Powell [89] proposes a derivative-free trust-region (DFTR) algorithm named COBYLA in which linear models are built by interpolating on non-degenerate simplices at each iteration. A merit function based on the infinity norm of the constraints allows to compute the trust region ratio. Originally proposed for inequality constraints, the software package NLOPT re-implements COBYLA and replaces each equality constraint by a pair of inequalities. Recently, Sampaio and Toint [96, 97] adapt the trust-funnel method in a DFTR algorithm to solve Problem (6.1). The trust-funnel method treats the equality constraint with no need of a filter, a barrier or a penalty [55].

Other algorithms provide treatment for problems close to (6.1) in a DFO context. A SQP derivative-free trust-region algorithm is proposed by Tröltzsch [102] for equality constrained problems. The NOWPAC algorithm [26] handles nonlinear inequality constraints in a DFTR algorithm, where strict feasibility is guaranteed at each iteration. Two recent papers deploy inexact restoration schemes. In [10] a violation measure is minimized in the restoration phase with a derivative-free algorithm, and a penalty-like merit function is exploited in the optimization phase. In [50], a filter is used in the optimization phase. Both papers uses models of the function in their optimization phase. The PBTR algorithm proposed in [14] considers problems with inequalities treated with the progressive barrier of [18] and uses IPOPT [103] to solve the underlying constrained subproblems. In [38] a DFTR algorithm treats Problem (6.1) using the gradients of the nonlinear constraints.

Many DFO algorithms use the augmented Lagrangian to handle constraints. The authors of [79] highlight a DFO issue : *"as this method requires both the objective function and constraints evaluations, it can be costly when the constraints can be easily evaluated without evaluating the objective function"*. This is not the case here, but it could be considered in future work. Torczon and Lewis [72] adapt the augmented Lagrangian algorithm [39] to a direct-search algorithm, without the use of derivatives. This adapted augmented Lagrangian is implemented in the direct-search HOPSPACK [74] method. The Algencan augmented Lagrangian method [29] is used in [48] to treat the difficult constraints whereas the easiest constraints are directly integrated into a subproblem solved by a DFO algorithm. An augmented Lagrangian method is also used in [104] where a DFTR algorithm is associated to a filter to solve problems with separable structure. Blackbox Optimization algorithms with surrogate models and augmented Lagrangian for inequalities and equalities are proposed in [56] and [86]. Finally in [9], augmented Lagrangian methods improve the solution of quadratic subproblems arising in the MADS direct-search algorithm.

Solving subproblems and trust region radius update are two important elements in the design of a DFTR algorithm for constrained problems. This paper proposes a new derivative-free trust-region algorithm called DFTR$^{\text{L}}$ treating general constrained problems by using an augmented Lagrangian.

The augmented Lagrangian is used at each iteration to solve a subproblem and the final augmented

Lagrangian function obtained at the end of the subproblem resolution is used to design simply the trust-region radius update rules. Section 6.2 presents a short review of the DFTR framework. Section 6.3 presents our new DFTR augmented Lagrangian algorithm for constraints, named DFTR$^{\mathrm{L}}$. Implementation details and computational results are exposed in Section 6.4. The proposed algorithm performs similarly to COBYLA on a set of analytical problems but outperforms COBYLA on problems with equalities and HOPSPACK on the entire chosen set of analytical problems. We conclude and evoke future work in Section 6.5.

6.2 A brief review of the DFTR framework

DFTR algorithms are inspired by the classical trust-region framework. Their originalities are the methods used to build the models and the subjacent theory guarantying similar convergence properties. As a trust-region algorithm [84, chap. 4], a DFTR algorithm solves a subproblem on a region where the original functions are replaced by models. These algorithms are efficient when the models of the original functions are good approximations within a trust region, a ball centered on the current iterate x^k, of radius Δ^k, the trust region radius at iteration k. Instead of solving the original problem, subproblems are iteratively optimized.

In classical trust-region methods with derivatives, the models are constructed using first or second order Taylor polynomials of the functions. In DFTR algorithms, the models of the functions cannot be built with the derivatives, since they are unavailable. Frequently used techniques to build models include interpolation or regression from a sample set of points around the current iterate. This sample set at iteration k is denoted by $\mathcal{Y}^k(x^k) \subset \mathbb{R}^n$.

Some properties are defined to characterize models offering similar properties than the first or second order truncated Taylor models based on derivatives. It is the case of the fully-linear models and the fully-quadratic models (see [43, chap. 6] for the formal definitions). Fully-linear models or fully-quadratic models can be guaranteed by some properties of the sample set. The well-poisedness is a geometric property characterizing a set of sample points. The theory is presented in [43, chap. 3]. If the sample set well-poisedness is satisfying, then an interpolated or regressed model computed from this sample set can be certifiably fully-linear or certifiably fully-quadratic. Some algorithms detailed in [43, chap. 6] explain how to construct such sample sets and models. From a given sample set it is also possible to improve the well-poisedness by replacing some points.

The stopping criteria is typically based on the radius Δ^k. Under certain assumptions the convergence analysis shows that the sequence of the trust region radii converges to zero, whereas in most trust-region algorithm the trust region radii diverge.

To summarize, DFTR is a trust-region algorithm with different mechanisms to build the models.

Thanks to new theories characterizing the sample set, we can certify to have Taylor-like models. Updates rules for the trust-region are simply adapted from the classical trust-region methods with derivatives. For more details, a basic unconstrained DFTR algorithm is presented in [43, chap. 10].

6.3 A DFTR algorithm using an augmented Lagrangian method

Augmented Lagrangian methods are a class of algorithms solving constrained nonlinear problems with derivatives. They belong to the class of penalty methods and use iteratively reformulated unconstrained problems thanks to an augmented Lagrangian function, which is the Lagrangian with an additional penalty term. Different augmented Lagrangian functions and algorithms exist. We use the augmented Lagrangian function defined by Powell, Hestenes and Rockafellar [59, 88, 94], called the PHR augmented Lagrangian. It is the one used in the Algencan algorithm detailed in [29] :

$$\mathcal{L}_\rho(x;\lambda,\mu) = f(x) + \frac{\rho}{2}\left(\sum_{i=1}^{m}\left[h_i(x)+\frac{\lambda_i}{\rho}\right]^2 + \sum_{i=1}^{p}\left[\max\left(0, g_i(x)+\frac{\mu_i}{\rho}\right)\right]^2\right),$$

where $\rho > 0$ is a penalty parameter. The coefficient $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^p_+$ are approximations of the Lagrange multipliers.

6.3.1 Solving the subproblems with an augmented Lagrangian method

As in the DFTR algorithm, our algorithm proposed in Section 6.3.2 solves a subproblem at iteration k within a trust region :

$$\begin{aligned} \min_{x\in B(x^k;\Delta^k)} \quad & \tilde{f}^k(x) \\ \text{subject to} \quad & \tilde{h}^k(x) = 0 \\ & \tilde{g}^k(x) \leq 0 \\ & l \leq x \leq u, \end{aligned} \tag{6.2}$$

where the functions $\tilde{f}^k$, $\tilde{h}^k$, and $\tilde{g}^k$ are quadratic models of f, h and g. The subproblems can be nonconvex with indefinite quadratic constraints, and are solved with an augmented Lagrangian algorithm at each iteration. The augmented Lagrangian of Problem (6.2) is :

$$\tilde{\mathcal{L}}^k_\rho(x;\lambda,\mu) = \tilde{f}^k(x) + \frac{\rho}{2}\left(\sum_{i=1}^{m}\left[\tilde{h}^k_i(x)+\frac{\lambda_i}{\rho}\right]^2 + \sum_{i=1}^{p}\left[\max\left(0, \tilde{g}^k_i(x)+\frac{\mu_i}{\rho}\right)\right]^2\right).$$

One can observe that the augmented Lagrangian of the subproblem, $\tilde{\mathcal{L}}_{\rho}^{k}$, is also a model of the augmented Lagrangian of the Problem (6.1), $\mathcal{L}_{\rho}$.

Birgin and Martinez [29] list advantages of using an augmented Lagrangian, and propose the Algencan algorithm, from which we borrowed the augmented Lagrangian to solve our subproblem. The principles of Algencan is to minimize at each iteration the unconstrained problem obtained with the augmented Lagrangian function with a precision ε^k satisfying $\varepsilon^k \to 0$. The penalty parameter ρ and the two coefficients λ and μ are updated at the end of each iteration. For example ρ is increased when the improvement is not sufficient regarding the feasibility of the new current point. This augmented Lagrangian algorithm always manages a current point x^k satisfying the bounds constraints. In the following we denote by $\tilde{\lambda}^k$, $\tilde{\mu}^k$, and $\tilde{\rho}^k$ the values of these coefficients at the end of the subproblem solution at iteration k of our DFTR algorithm. The augmented Lagrangian function at iteration k after solving the subproblem is denoted by $\mathcal{L}_{\tilde{\rho}^k}(x; \tilde{\lambda}^k, \tilde{\mu}^k)$, whereas the current augmented Lagrangian model function is denoted by $\tilde{\mathcal{L}}_{\tilde{\rho}^k}(x; \tilde{\lambda}^k, \tilde{\mu}^k)$.

6.3.2 A DFTR algorithm based on the augmented Lagrangian

The current augmented Lagrangian function and the current augmented Lagrangian model function are used to compute the trust region ratio r^k, measuring the quality of the minimization of the original problem in comparison with the expected minimization obtained with the subproblem. We denote by $\tilde{x}^k$ the solution of the model subproblem :

$$r^k = \frac{\mathcal{L}_{\tilde{\rho}^k}(x^k; \tilde{\lambda}^k, \tilde{\mu}^k) - \mathcal{L}_{\tilde{\rho}^k}(\tilde{x}^k; \tilde{\lambda}^k, \tilde{\mu}^k)}{\tilde{\mathcal{L}}_{\tilde{\rho}^k}(x; \tilde{\lambda}^k, \tilde{\mu}^k) - \tilde{\mathcal{L}}_{\tilde{\rho}^k}(\tilde{x}^k; \tilde{\lambda}^k, \tilde{\mu}^k)}.$$

The new algorithm named DFTR$^{\mathrm{L}}$ is outlined in Figure 2. The algorithm parameters η_0, η_1, γ_{inc}, and γ_{dec} must respect the following conditions : $0 \leq \eta_0 < \eta_1 < 1$, $0 < \gamma_{inc} < 1 < \gamma_{dec}$. The parameters η_0 and η_1 are thresholds to quantify the quality of the ratio r^k. The parameters γ_{inc} and γ_{dec} are coefficients to increase or decrease the trust region radius Δ^k based on the quality of the ratio r^k.

6.4 Implementation details and computational results

Our algorithm DFTR$^{\mathrm{L}}$ is implemented in Python and this section compares it with two state-of-the-art software packages. We first describe our set of analytical problems and then the tools to analyse the results.

Algorithm 2 Algorithm DFTR$^{\mathrm{L}}$: iteration k.

Model construction
Construct the set of sample points $\mathcal{Y}^k$ around x^k and build the models $\tilde{f}^k$, $\tilde{h}^k$, $\tilde{g}^k$.
Subproblem solution
Solve Subproblem (6.2)

$$\begin{aligned} \min_{x \in B(x^k;\Delta^k)} \quad & \tilde{f}^k(x) \\ \text{subject to} \quad & \tilde{h}^k(x) = 0 \\ & \tilde{g}^k(x) \leq 0 \\ & l \leq x \leq u, \end{aligned}$$

within the trust region with the augmented Lagrangian algorithm. The algorithm returns $\tilde{x}^k$, $\tilde{\lambda}^k$, $\tilde{\mu}^k$ and $\tilde{\rho}^k$.
Step calculation
Evaluate f, h, and g at $\tilde{x}^k$ and compute the ratio r^k at $\tilde{x}^k$.
Trust region radius update
If $r^k \geq \eta_1$, then set $x^{k+1} = \tilde{x}^k$ and $\Delta^{k+1} = \min\left(\gamma_{inc}\Delta^k, \Delta_{max}\right)$.
If $\eta_0 \leq r^k < \eta_1$, then set $x^{k+1} = \tilde{x}^k$ and $\Delta^{k+1} = \Delta^k$.
If $r^k < \eta_0$, then set $x^{k+1} = x^k$ and $\Delta^{k+1} = \gamma_{dec}\Delta^k$.

6.4.1 Computational testbed

We used a set of 83 small-scale analytical problems from the CUTEst collection [54]. This set includes the test problems used in [14]. Among them, 40 contain only inequality constraints. Their characteristics are presented in appendix. The initial point proposed in the CUTEst collection satisfies the bound constraints. A budget of 100(n+1) blackbox evaluations is chosen.

COBYLA is a DFTR algorithm using a l_∞ merit function and linear models, and HOPSPACK is a direct-search based method using an augmented Lagrangian to treat general constraints.

These algorithms are used to analyse the performance of our algorithm because COBYLA is also a DFTR algorithm using a merit function and HOPSPACK is a direct-search method using an augmented Lagrangian algorithm. We use the NLOPT version of COBYLA with default settings. HOPSPACK is used with default parameters and a tolerance of 10^{-7} for each constraint. Note that HOPSPACK allows an explicit treatment of linear constraints, and as neither COBYLA nor our implementation contains this feature, it has been disabled in order to allow a fair comparison.

Data profiles and performance profiles from [83] are used to analyze performance. These graphs compare different algorithms on a given set of problems. For a tolerance parameter $\tau \in [0; 1]$, fixed to 10^{-3} in this paper, data profiles present, for a particular budget of evaluations, the percentage of problems providing a solution within τ to a reference equal to the best solution found by all the algorithms. When no feasible solution has been found, no algorithm is considered to have solved this problem. A point is considered feasible when every constraint is satisfied within a tolerance of 10^{-7}.

Performance profiles from [83] are also used. A performance ratio $r_{p,s}$ is defined by

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}$$

for Algorithm s on Problem p where S is the set of algorithms tested. If for example $r_{p,s} = 2$, Algorithm s needs twice the number of evaluations of the best algorithm to solve Problem p, within a tolerance τ. The performance profiles show for $\alpha \geq 1$ the fraction of problems solved by Algorithm s with a ratio $r \leq \alpha$. The value of a performance profile for $\alpha = 1$ indicates the proportion of problems a given algorithm solves the best (two algorithms can equally solve one problem), and a performance profile when $\alpha \to \infty$ indicates the proportion of problems efficiently solved by the algorithm.

The sample set used to build the quadratic interpolation models requires at each iteration $(n+1)(n+2)/2$ points. These points correspond to the most recent points in a ball of radius $2\Delta^k$ around the current iterate x^k. If there are not enough points, then the geometry improvement algorithm is called

to select new points by keeping a well-poised geometry of the sample set.

The subproblems are optimized with the Algencan algorithm implemented in the NLOPT package. A limit of 5000 iterations is imposed, and the subproblem tolerance for each constraint is 10^{-8}. The original problem tolerance for each constraint is 10^{-7}.

6.4.2 Comparison with COBYLA and HOPSPACK

Our algorithm is compared to the two state-of-the art software packages COBYLA and HOPSPACK. The results are presented separately for constrained problems without equalities and those with at least one equality.

This distinction is made to analyze the performance of our algorithm on different kinds of problems.

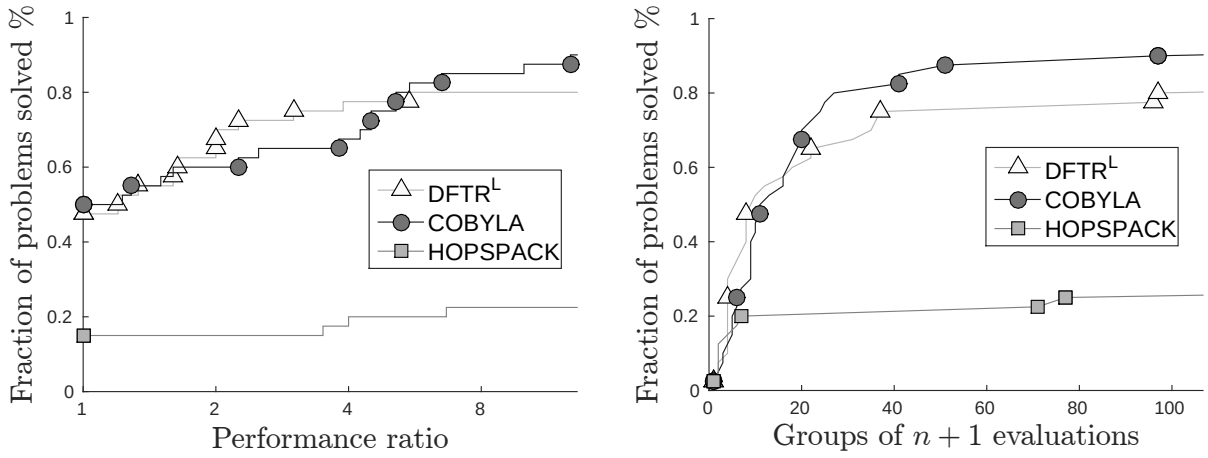
Inequality constrained problems. The performance profiles in Figure 6.1(a) show that our algorithm is competitive with COBYLA on the benchmark set of 40 inequality constrained CUTEst problems. Both DFTR$^{\mathrm{L}}$ and COBYLA perform better than the direct-search HOPSPACK algorithm using augmented Lagrangian method. The performance of DFTR$^{\mathrm{L}}$ is comparable to that of COBYLA even if DFTR$^{\mathrm{L}}$ is slightly below. The performance profiles show that DFTR$^{\mathrm{L}}$ solves 10% fewer inequality constrained problems than COBYLA. The data profiles in Figure 6.1(b) confirm these observations : Even if DFTR$^{\mathrm{L}}$ seems a bit faster when the number of function evaluations is above $20(n+1)$, COBYLA outperforms DFTR$^{\mathrm{L}}$ on 10% of the tested problems with a larger number of evaluations. These results show that DFTR$^{\mathrm{L}}$ is competitive with COBYLA but slightly less efficient.

General constrained problems with at least one equality. Computational results with 43 problems containing at least one equality show that DFTR$^{\mathrm{L}}$ globally outperforms COBYLA on problems with at least one equality. The performance profiles in Figure 6.2(a) show that our algorithm solves more than 20% of the problems faster than COBYLA, and is able to asymptotically solves almost 10% more. Both DFTR$^{\mathrm{L}}$ and COBYLA dominate the direct-search HOPSPACK algorithm.

HOPSPACK is less efficient as it is a direct-search method which does not exploit the curvature information. The data profiles in Figure 6.2(b) confirm these observations. The performance of DFTR$^{\mathrm{L}}$ is comparable to that of COBYLA, and DFTR$^{\mathrm{L}}$ outperforms COBYLA slightly.

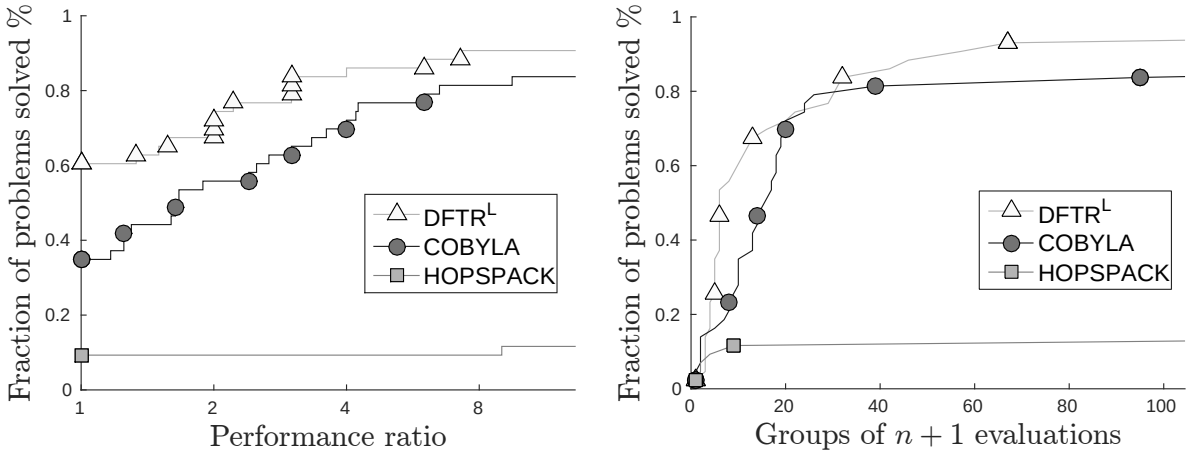
6.5 Discussion

This work proposes a derivative-free trust-region algorithm to treat general nonlinear cons-traints for problems without the use of their derivatives. The augmented Lagrangian method and func-



(a) Perf. profiles at $100(n+1)$ evaluations. (b) Data profiles with $\tau = 10^{-3}$.

Figure 6.1 Comparison of DFTR$^{\mathrm{L}}$ with COBYLA and HOPSPACK on analytical CUTEst problems with only inequalities.



(a) Perf. profiles at $100(n+1)$ evaluations. (b) Data profiles with $\tau = 10^{-3}$.

Figure 6.2 Comparison of DFTR$^{\mathrm{L}}$ with COBYLA and HOPSPACK on analytical CUTEst problems with at least one equality.

tion are used to both solve the trust-region subproblem and simply design the update rules of the derivative-free trust-region algorithm, with few modifications to the unconstrained DFTR framework.

Computational experiments are conducted on a collection of 80 problems from the CUTEst collection with two state-of-the art algorithms : HOPSPACK, a direct-search algorithm using an augmented Lagrangian method for the constraints, and COBYLA, a DFTR algorithm. Our new algorithm, DFTR^{L}, outperforms HOPSPACK and is competitive with COBYLA on analytical problems. It is worth noting that DFTR^{L} performs better on problems with equalities.

Future work may consider other penalty functions to solve the subproblem and design the trust region ratio. Other sample set managements could be tested to improve the performance. Finally, the progressive barrier [18] could be adapted to this new algorithm to improve the treatment of inequalities.

6.6 Appendix

Table 6.1 Description of the 40 analytical problems with only inequalities ($m = 0$).

Name	n	p	lower bounds	upper bounds	initial point
avgasb	8	10	8	8	Feasible
b2	3	3	0	0	Infeasible
chaconn1	3	3	0	0	Infeasible
himmelp5	2	3	2	2	Infeasible
hs10	2	1	0	0	Infeasible
hs11	2	1	0	0	Infeasible
hs12	2	1	0	0	Feasible
hs15	2	2	0	1	Infeasible
hs18	2	2	2	2	Infeasible
hs19	2	2	2	2	Infeasible
hs22	2	2	0	0	Infeasible
hs23	2	5	2	2	Infeasible
hs24	2	3	2	0	Feasible
hs29	3	1	0	0	Feasible
hs30	3	1	3	3	Feasible
hs31	3	1	3	3	Feasible
hs33	3	2	3	1	Feasible
hs34	3	2	3	3	Feasible
hs35	3	1	3	0	Feasible
hs36	3	1	3	3	Feasible
hs43	4	3	0	0	Feasible
hs57	2	1	2	0	Feasible
hs64	3	1	3	0	Infeasible
hs72	4	2	4	4	Infeasible
hs76	4	3	4	0	Feasible
hs84	5	6	5	5	Feasible
hs86	5	10	5	0	Feasible
hs95	6	4	6	6	Infeasible
hs96	6	4	6	6	Infeasible
hs97	6	4	6	6	Infeasible
hs98	6	4	6	6	Infeasible
hs100	7	4	0	0	Feasible
hs101	7	6	7	7	Infeasible
hs108	9	13	1	0	Infeasible
kiwcresc	3	2	0	0	Infeasible
lootsma	3	2	0	1	Feasible
polak6	5	4	0	0	Infeasible
simpllpb	2	3	0	0	Infeasible
snake	2	2	0	0	Infeasible
spiral	3	2	0	0	Feasible

Table 6.2 Description of the 43 analytical problems with at least one equality constraint.

Name	n	m	p	lower bounds	upper bounds	initial point
booth	2	2	0	0	0	Infeasible
bt4	3	2	0	0	0	Infeasible
bt5	3	2	0	0	0	Infeasible
bt8	5	2	0	0	0	Infeasible
bt13	5	1	0	0	1	Infeasible
byrdsphr	3	2	0	0	0	Infeasible
cluster	2	2	0	0	0	Infeasible
dixchlng	10	5	0	0	0	Infeasible
extrasim	2	1	0	0	2	Infeasible
gottfr	2	2	0	0	0	Infeasible
hs006	2	1	0	0	0	Infeasible
hs007	2	1	0	0	0	Infeasible
hs008	2	2	0	0	0	infeasible
hs014	2	1	1	0	0	Infeasible
hs027	3	1	0	0	0	Infeasible
hs028	3	1	0	0	0	Feasible
hs032	3	1	1	3	0	Feasible
hs039	4	2	0	0	0	Infeasible
hs040	4	3	0	0	0	Infeasible
hs042	4	0	0	0	0	Infeasible
hs048	5	2	0	0	0	Feasible
hs052	5	3	0	0	0	Infeasible
hs053	5	3	5	5	5	Infeasible
hs054	6	1	0	6	6	Infeasible
hs055	6	6	0	6	2	Infeasible
hs060	3	1	0	3	3	Infeasible
hs061	3	2	0	0	0	Infeasible
hs062	3	1	0	3	3	Feasible
hs063	3	2	0	3	0	Infeasible
hs071	4	1	1	4	40	Feasible
hs073	4	1	2	4	0	Infeasible
hs078	5	3	0	0	0	Infeasible
hs080	5	3	0	5	5	Infeasible
hs111	10	3	0	10	10	Infeasible
hs112	10	3	0	10	0	Infeasible
hs114	10	3	8	0	0	Infeasible
hypcir	2	2	0	0	0	Infeasible
maratos	2	1	0	0	0	Infeasible
odfits	10	6	0	10	0	Infeasible
portfl1	12	1	0	12	12	Feasible
supersim	2	2	0	2	0	Infeasible
tame	2	1	0	2	0	Infeasible
zangwil3	3	3	0	0	0	Infeasible

CHAPITRE 7 DISCUSSION GÉNÉRALE

7.1 Synthèse des travaux

Cette thèse a permis l'élaboration de trois nouvelles techniques pour résoudre des problèmes DFO et BBO sous contraintes.

La première technique, proposée dans un cadre BBO, mais utilisable telle quelle dans un cadre DFO, permet de traiter les contraintes d'égalités linéaires en réduisant le degré de liberté du problème initial. L'analyse de convergence sur le problème réduit dans le cadre BBO avec l'utilisation de l'algorithme MADS est similaire à celle de l'algorithme MADS sans contraintes d'égalités linéaires. Les résultats numériques obtenus améliorent ceux de HOPSPACK.

La deuxième technique proposée est une adaptation dans un algorithme DFTR de la barrière progressive. Les contraintes d'inégalités de simulation relaxables et quantifiables sont modélisées de la même manière que la fonction objectif par des modèles linéaires ou quadratiques. Deux sous-problèmes utilisant les fonctions modèles sont résolus à chaque itération, et deux rayons de confiance sont maintenus. Une analyse de convergence permet de conclure que l'algorithme converge, puisque au moins un des deux rayons de confiance converge vers 0. Les résultats obtenus sont compétitifs avec les logiciels COBYLA et NOMAD, montrant la validité de l'adaptation de la barrière progressive dans un algorithme DFO.

Enfin, un dernier algorithme est proposé, tirant profit d'une méthode de Lagrangien augmenté, à la fois pour résoudre le sous-problème mais aussi pour calculer le ratio servant à mettre à jour le rayon de confiance avec des règles minimalistes. L'algorithme implémenté est compétitif avec COBYLA, et améliore le traitement des problèmes avec contraintes d'égalités.

7.2 Limitations des solutions proposées

La solution apportée par le premier projet traitant les contraintes d'égalités linéaires n'a pas été testée avec un autre algorithme BBO ou DFO que MADS. De plus, elle n'a pas été comparée au logiciel LINCOA qui a été développé en parallèle de la réalisation de ce projet.

Les algorithmes DFTR implémentés dans les deux derniers projets sont efficaces. Cependant aucune expérimentation n'a été faite pour améliorer leur performance vis-à-vis de la gestion des ensembles de points servant à calculer les modèles. En effet, l'emphase a été mise sur la simplicité des algorithmes développés au détriment parfois de leur efficacité.

De plus, les algorithmes proposés construisent des modèles linéaires ou quadratiques. Et contrai-

rement aux attentes, les algorithmes avec des modèles quadratiques ne sont pas considérablement meilleurs que les algorithmes avec des modèles linéaires.

Enfin, seule une analyse de convergence très simple est proposée dans le cadre des projets impliquant les algorithmes DFTR.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Le traitement des contraintes d'égalités linéaires pourra être intégrée au logiciel NOMAD. De plus, un algorithme présentant une technique de gestion des contraintes d'égalité générale est proposée dans [33] à partir d'un algorithme BBO, à savoir HOPSPACK, permettant le traitement explicite des contraintes Comme notre traitement des contraintes d'égalités linéaires testé dans NOMAD semble surpasser HOPSPACK, il pourrait être intéressant de tester l'algorithme de restoration avec le nouvel algorithme que nous avons obtenu, à savoir MADS pour les contraintes d'égalités linéaires.

Concernant les deux derniers projets impliquant des algorithmes DFTR, les améliorations les plus importantes concernent la gestion des ensembles de points servant à construire les modèles. Des techniques plus économiques devraient être testées, visant à construire davantage de modèles sous-déterminés. De plus, la géométrie des ensembles de points servant à construire les modèles ne doit pas nécessairement être de bonne qualité à chaque itération, mais seulement lorsque l'on veut réduire la région de confiance ou lorsque l'on est proche d'un point critique pour les modèles. Une comparaison de différentes stratégies de gestion de ces ensembles et des modèles pourraient être menées dans le cadre des nouveaux algorithmes que nous avons proposés. Une étude plus poussée, d'abord dans le cas sans contraintes, du choix linéaire ou quadratique des modèles mériterait aussi une attention particulière. L'amélioration des analyses de convergence afin d'obtenir davantage de résultats théoriques est aussi une piste pour le futur.

Nous avons montré la pertinence de l'adaptation de la barrière progressive, une technique BBO, dans un algorithme DFO. La barrière progressive pourrait aussi être testée dans un algorithme classique d'optimisation avec dérivées. De la même manière, davantage de techniques issues des domaines de l'optimisation avec dérivées pourraient être adaptées à des algorithmes DFO voire BBO, afin d'améliorer leurs performances.

RÉFÉRENCES

[1] M.A. Abramson. Mixed variable optimization of a Load-Bearing thermal insulation system using a filter pattern search algorithm. *Optimization and Engineering*, 5(2) :157–177, 2004.

[2] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, Jr., S. Le Digabel, and C. Tribes. The NOMAD project. Software available at `https://www.gerad.ca/nomad`, 2008.

[3] M.A. Abramson, C. Audet, and J.E. Dennis, Jr. Generalized pattern searches with derivative information. *Mathematical Programming*, Series B, 100(1) :3–25, 2004.

[4] M.A. Abramson, C. Audet, and J.E. Dennis, Jr. Filter pattern search algorithms for mixed variable constrained optimization problems. *Pacific Journal of Optimization*, 3(3) :477–500, 2007.

[5] M.A. Abramson, C. Audet, J.E. Dennis, Jr., and S. Le Digabel. OrthoMADS : A Deterministic MADS Instance with Orthogonal Directions. *SIAM Journal on Optimization*, 20(2) :948–966, 2009.

[6] M.A. Abramson, O.A. Brezhneva, J.E. Dennis Jr., and R.L. Pingel. Pattern search in the presence of degenerate linear constraints. *Optimization Methods and Software*, 23(3) :297–319, 2008.

[7] S. Alarie, C. Audet, V. Garnier, S. Le Digabel, and L.-A. Leclaire. Snow water equivalent estimation using blackbox optimization. *Pacific Journal of Optimization*, 9(1) :1–21, 2013.

[8] P. Alberto, F. Nogueira, H. Rocha, and L.N. Vicente. Pattern search methods for user-provided points : Application to molecular geometry problems. *SIAM Journal on Optimization*, 14(4) :1216–1236, 2004.

[9] N. Amaioua, C. Audet, A. R. Conn, and S. Le Digabel. Efficient solution of quadratically constrained quadratic subproblems within a direct-search algorithm. Technical Report G-2016-45, Les cahiers du GERAD, 2016.

[10] M.B. Arouxét, N.E. Echebest, and E.A. Pilotta. Inexact Restoration method for nonlinear optimization without derivatives. *Journal of Computational and Applied Mathematics*, 290 :26–43, 2015.

[11] C. Audet. Convergence Results for Generalized Pattern Search Algorithms are Tight. *Optimization and Engineering*, 5(2) :101–122, 2004.

[12] C. Audet. A survey on direct search methods for blackbox optimization and their applications. In P.M. Pardalos and T.M. Rassias, editors, *Mathematics without boundaries : Surveys in interdisciplinary research*, chapter 2, pages 31–56. Springer, 2014.

[13] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2) :299–318, 2008.

[14] C. Audet, A. R. Conn, S. Le Digabel, and M. Peyrega. A progressive barrier derivative-free trust-region algorithm for constrained optimization. Technical Report G-2016-49, Les cahiers du GERAD, 2016.

[15] C. Audet and J.E. Dennis, Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3) :889–903, 2003.

[16] C. Audet and J.E. Dennis, Jr. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, 14(4) :980–1010, 2004.

[17] C. Audet and J.E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1) :188–217, 2006.

[18] C. Audet and J.E. Dennis, Jr. A Progressive Barrier for Derivative-Free Nonlinear Programming. *SIAM Journal on Optimization*, 20(1) :445–472, 2009.

[19] C. Audet, J.E. Dennis, Jr., and S. Le Digabel. Globalization strategies for Mesh Adaptive Direct Search. *Computational Optimization and Applications*, 46(2) :193–215, 2010.

[20] C. Audet, K. Diest, S. Le Digabel, L.A. Sweatlock, and D.E. Marthaler. Metamaterial Design by Mesh Adaptive Direct Search. In *Numerical Methods for Metamaterial Design*, volume 127 of *Topics in Applied Physics*, pages 71–96. Springer, 2013.

[21] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. En préparation, prévu pour 2017.

[22] C. Audet, A. Ianni, S. Le Digabel, and C. Tribes. Reducing the Number of Function Evaluations in Mesh Adaptive Direct Search Algorithms. *SIAM Journal on Optimization*, 24(2) :621–642, 2014.

[23] C. Audet and M. Kokkolaras. Blackbox and derivative-free optimization : theory, algorithms and applications. *Optimization and Engineering*, 17(1) :1–2, 2016.

[24] C. Audet, S. Le Digabel, and M. Peyrega. Linear equalities in blackbox optimization. *Computational Optimization and Applications*, 61(1) :1–23, 2015.

[25] C. Audet, S. Le Digabel, and M. Peyrega. A derivative-free trust-region augmented Lagrangian algorithm. Technical Report G-2016-xx, Les cahiers du GERAD, 2016.

[26] F. Augustin and Y.M. Marzouk. NOWPAC : A provably convergent derivative-free nonlinear optimizer with path-augmented constraints. Technical report, Massachusetts Institute of Technology, 2014.

[27] A.S. Bandeira, K. Scheinberg, and L.N. Vicente. Convergence of trust-region methods based on probabilistic models. *SIAM Journal on Optimization*, 24(3) :1238–1264, 2014.

[28] D. P. Bertsekas. *Constrained Optimization and Lagrangian Multiplier Methods*. Academic Press, New York, 1982.

[29] E. G. Birgin and J. M. Martinez. Improving ultimate convergence of an augmented lagrangian method, 2007.

[30] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.W. Moore, and D.B. Serafini. Managing surrogate objectives to optimize a helicopter rotor design – further experiments. AIAA Paper –4717, Presented at the 8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, 1998.

[31] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, and V. Torczon. Optimization using surrogate objectives on a helicopter test example. In J. Borggaard, J. Burns, E. Cliff, and S. Schreck, editors, *Optimal Design and Control*, Progress in Systems and Control Theory, pages 49–58, Cambridge, Massachusetts, 1998. Birkhäuser.

[32] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. *Structural and Multidisciplinary Optimization*, 17(1) :1–13, 1999.

[33] L.F. Bueno, A. Friedlander, J.M. Martínez, and F.N.C. Sobral. Inexact restoration method for derivative-free optimization with smooth constraints. *SIAM Journal on Optimization*, 23(2) :1189–1213, 2013.

[34] R.J. Hanson C. L. Lawson. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[35] A. Çivril and M. Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47–49) :4801–4811, 2009.

[36] T.D. Choi and C.T. Kelley. Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4) :1149–1162, 2000.

[37] F.H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[38] P.D. Conejo, E.W. Karas, and L.G. Pedroso. A trust-region derivative-free algorithm for constrained optimization. *Optimization Methods and Software*, 30(6) :1126–1145, 2015.

[39] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2) :545–572, 1991.

[40] A.R. Conn and S. Le Digabel. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, 28(1) :139–158, 2013.

[41] A.R. Conn, K. Scheinberg, and Ph.L. Toint. A derivative free optimization algorithm in practice. In *Proceedings the of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Missouri, 1998.

[42] A.R Conn, K. Scheinberg, and L.N. Vicente. Global convergence of general derivative-free trust-region algorithms to first and second order critical points. *SIAM Journal on Optimization*, 20(1) :387–415, 2009.

[43] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[44] I.D. Coope and C.J. Price. Frame-based methods for unconstrained optimization. *Journal of Optimization Theory and Applications*, 107(2) :261–274, 2000.

[45] I.D. Coope and C.J. Price. On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization*, 11(4) :859–869, 2001.

[46] C. Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76 :733–746, 1954.

[47] J.E. Dennis, Jr., C.J. Price, and I.D. Coope. Direct Search Methods for Nonlinearly Constrained Optimization Using Filters and Frames. *Optimization and Engineering*, 5(2) :123–144, 2004.

[48] M.A. Diniz-Ehrhardt, J.M. Martinez, and L.G. Pedroso. Derivative-free methods for nonlinear programming with general lower-level constraints. *Computational and Applied Mathematics*, 30 :19–52, 2011.

[49] D.W. Dreisigmeyer. Equality constraints, Riemannian manifolds and direct search methods. Technical Report LA-UR-06-7406, Los Alamos National Laboratory, Los Alamos, USA, 2006.

[50] N. Echebest, M.L. Schuverdt, and R.P. Vignau. An inexact restoration derivative-free filter method for nonlinear programming. *Computational and Applied Mathematics*, pages 1–26, 2015.

[51] E. Fermi and N. Metropolis. Technical report, Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA, 1952.

[52] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, Series A, 91 :239–269, 2002.

[53] K.R. Fowler, C.T. Kelley, C.T. Miller, C.E. Kees, R.W. Darwin, J.P. Reese, M.W. Farthing, and M.S.C. Reed. Solution of a well-field design problem with implicit filtering. *Optimization and Engineering*, 5(2) :207–234, 2004.

[54] N.I.M. Gould, D. Orban, and Ph.L. Toint. CUTEst : a Constrained and Unconstrained Testing Environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3) :545–557, 2015. Code available at `http://ccpforge.cse.rl.ac.uk/gf/project/cutest/wiki`.

[55] N.I.M. Gould and Ph.L. Toint. Nonlinear programming without a penalty function or a filter. *Mathematical Programming*, 122(1) :155–196, 2010.

[56] R.B. Gramacy, G.A. Gray, S. Le Digabel, H.K.H. Lee, P. Ranjan, G. Wells, and S.M. Wild. Modeling an Augmented Lagrangian for Blackbox Constrained Optimization. *Technometrics*, 58(1) :1–11, 2016.

[57] J.D. Griffin, T.G. Kolda, and R.M. Lewis. Asynchronous parallel generating set search for linearly-constrained optimization. *SIAM Journal on Scientific Computing*, 30(4) :1892–1924, 2008.

[58] E.A.E Gumma, M.H.A. Hashim, and M. Montaz Ali. A derivative-free algorithm for linearly constrained optimization problems. *Computational Optimization and Applications*, 57(3) :599–621, 2014.

[59] M.R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5) :303–320, 1969.

[60] W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*, volume 187 of *Lecture Notes in Economics and Mathematical Systems*. Springer Verlag, Berlin, Germany, 1981.

[61] B.B. Ivanov, A.A. Galushko, and R.P. Stateva. Phase stability analysis with equations of state – a fresh look from a different perspective. *Industrial & Engineering Chemistry Research*, 52(32) :11208–11223, 2013.

[62] J. Jahn. *Introduction to the Theory of Nonlinear Optimization*. Springer, Berlin, 1994.

[63] M. Kokkolaras, C. Audet, and J.E. Dennis, Jr. Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Optimization and Engineering*, 2(1) :5–29, 2001.

[64] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by direct search : New perspectives on some classical and modern methods. *SIAM Review*, 45(3) :385–482, 2003.

[65] T.G. Kolda, R.M. Lewis, and V. Torczon. A generating set direct search augmented Lagrangian algorithm for optimization with a combination of general and linear constraints. Technical Report SAND2006-5315, Sandia National Laboratories, USA, 2006.

[66] T.G. Kolda, R.M. Lewis, and V. Torczon. Stationarity results for generating set search for linearly constrained optimization. *SIAM Journal on Optimization*, 17(4) :943–968, 2006.

[67] S. Le Digabel. Algorithm 909 : NOMAD : Nonlinear Optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4) :44 :1–44 :15, 2011.

[68] S. Le Digabel and S.M. Wild. A Taxonomy of Constraints in Simulation-Based Optimization. Technical Report G-2015-57, Les cahiers du GERAD, 2015.

[69] R.M. Lewis, A. Shepherd, and V. Torczon. Implementing Generating Set Search Methods for Linearly Constrained Minimization. *SIAM Journal on Scientific Computing*, 29(6) :2507–2530, 2007.

[70] R.M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4) :1082–1099, 1999.

[71] R.M. Lewis and V. Torczon. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3) :917–941, 2000.

[72] R.M. Lewis and V. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4) :1075–1089, 2002.

[73] R.M. Lewis and V. Torczon. Active set identification for linearly constrained minimization without explicit derivatives. *SIAM Journal on Optimization*, 20(3) :1378–1405, 2009.

[74] R.M. Lewis and V. Torczon. A direct search approach to nonlinear programming problems using an augmented lagrangian method with explicit treatment of linear constraints. Technical report, College of William & Mary, 2010.

[75] L. Liu and X. Zhang. Generalized pattern search methods for linearly equality constrained optimization problems. *Applied Mathematics and Computation*, 181(1) :527–535, 2006.

[76] G. Liuzzi and S. Lucidi. A derivative-free algorithm for inequality constrained nonlinear programming via smoothing of an ℓ_∞ penalty function. *SIAM Journal on Optimization*, 20(1) :1–29, 2009.

[77] G. Liuzzi, S. Lucidi, and M. Sciandrone. Sequential penalty derivative-free methods for nonlinear constrained optimization. *SIAM Journal on Optimization*, 20(5) :2614–2635, 2010.

[78] A.L. Marsden, M. Wang, J.E. Dennis, Jr., and P. Moin. Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering*, 5(2) :235–262, 2004.

[79] J.M. Martínez and F.N.C Sobral. Constrained derivative-free optimization on thin domains. *Journal of Global Optimization*, 56(3) :1217–1232, 2013.

[80] J.H. May. *Linearly Constrained Nonlinear Programming : A Solution Method That Does Not Require Analytic Derivatives*. PhD thesis, Yale University, December 1974.

[81] R. Mifflin. A superlinearly convergent algorithm for minimization without evaluating derivatives. *Mathematical Programming*, 9(1) :100–117, 1975.

[82] M. Minville, D. Cartier, C. Guay, L.-A. Leclaire, C. Audet, S. Le Digabel, and J. Merleau. Improving process representation in conceptual hydrological model calibration using climate simulations. *Water Resources Research*, 50 :5044–5073, 2014.

[83] J.J. Moré and S.M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1) :172–191, 2009.

[84] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 1999.

[85] R. Perez, H.H.T. Liu, and K. Behdinan. Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, September 2004.

[86] V. Picheny, R.B. Gramacy, S.M. Wild, and S. Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. Technical Report G-2016-43, Les cahiers du GERAD, 2016.

[87] T.D. Plantenga. HOPSPACK 2.0 User Manual. Technical Report SAND2009-6265, Sandia National Laboratories, Livermore, CA, October 2009.

[88] M.J.D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, 1969.

[89] M.J.D. Powell. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In S. Gomez and J.-P. Hennart, editors, *Advances in Optimization and Numerical Analysis*, volume 275 of *Mathematics and Its Applications*, pages 51–67. Springer Netherlands, 1994.

[90] M.J.D. Powell. Lincoa software. Software available at `http://mat.uc.pt/~zhang/software.html#lincoa`, 2014.

[91] M.J.D. Powell. On fast trust region methods for quadratic models with linear constraints. Technical Report DAMTP 2014/NA02, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Silver Street, Cambridge CB3 9EW, England, 2014.

[92] M.J.D. Powell. On fast trust region methods for quadratic models with linear constraints. *Mathematical Programming Computation*, 7(3) :237–267, 2015.

[93] C.J. Price and I.D. Coope. Frames and grids in unconstrained and linearly constrained optimization : A nonsmooth approach. *SIAM Journal on Optimization*, 14(2) :415–438, 2003.

[94] R. T. Rockafellar. Augmented lagrange multiplier functions and duality in nonconvex programming. *SIAM Journal on Control*, 12(2) :268–285, 1974.

[95] W. Rudin. *Functional Analysis*. International Series in Pure and Applied Mathematics. McGraw-Hill Inc., New York, second edition, 1991.

[96] Ph.R. Sampaio and Ph.L. Toint. A derivative-free trust-funnel method for equality-constrained nonlinear optimization. *Computational Optimization and Applications*, 61(1) :25–49, 2015.

[97] Ph.R. Sampaio and Ph.L. Toint. Numerical experience with a derivative-free trust-funnel method for nonlinear optimization problems with general nonlinear constraints. *Optimization Methods and Software*, 31(3) :511–534, 2016.

[98] S.E. Selvan, P.B. Borckmans, A. Chattopadhyay, and P.-A. Absil. Spherical mesh adaptive direct search for separating quasi-uncorrelated sources by range-based independent component analysis. *Neural Computation*, 25(9) :2486–2522, 2013.

[99] J. Sobieszczanski-Sobieski, J.S. Agte, and R.R. Sandusky, Jr. Bilevel Integrated System Synthesis. *AIAA Journal*, 38(1) :164–172, 2000.

[100] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1) :1–25, 1997.

[101] C. Tribes, J.-F. Dubé, and J.-Y. Trépanier. Decomposition of multidisciplinary optimization problems : formulations and application to a simplified wing design. *Engineering Optimization*, 37(8) :775–796, 2005.

[102] A. Tröltzsch. A sequential quadratic programming algorithm for equality-constrained optimization without derivatives. *Optimization Letters*, 10(2) :383–399, 2016.

[103] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1) :25–57, 2006.

[104] D. Xue and W. Sun. On convergence analysis of a derivative-free trust region algorithm for constrained optimization with separable structure. *Science China Mathematics*, 57(6) :1287–1302, 2014.

[105] Y. Yuan. Recent advances in trust region algorithms. *Mathematical Programming*, 151(1) :249–281, 2015.

[106] Y.-X. Yuan. An example of non-convergence of trust region algorithms. In Y.-X. Yuan, editor, *Advances in Nonlinear Programming*, pages 205–215. Kluwer Academic, Dordercht, 1998.